Implementación de prácticas de seguridad en un pipeline de entrega continua para entornos Salesforce

José María González Sanz

Máster Universitario en Ciberseguridad y Privacidad
Seguridad Empresarial

Nombre Tutor/a de TF
Miguel Ángel Flores Terrón
Profesor/a responsable de la asignatura
Víctor García Font

Fecha Entrega
06/2024

Universitat Oberta de Catalunya

GNU Free Documentation License (GNU FDL)

Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación de prácticas de seguridad en un pipeline de entrega continua para entornos Salesforce*
Nombre del autor:	*José María González Sanz*
Nombre del consultor/a:	*Miguel Ángel Flores Terrón*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*06/2024*
Titulación o programa:	*Máster Universitario en Ciberseguridad y Privacidad*
Área del Trabajo Final:	*Seguridad Empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*DevSevOps, CI/CD Pipeline, Salesforce*

Resumen del Trabajo

La necesidad de entregar soluciones de software garantizando la seguridad del producto y de los procesos asociados es cada vez más importante debido al incremento de las amenazas sobre los entornos digitales. El propósito de este trabajo es crear un pipeline de entrega continúa para soluciones Salesforce aplicando DevSecOps.

Con este objetivo, se consideran las fases del ciclo de vida que aplican a las soluciones de Salesforce. Para cada una de estas fases, se analizan y comparan diversas herramientas con tal de elegir la que se adecue mejor a la tarea. Tras el análisis de las herramientas, se procede a proponer un diseño de pipeline. Utilizando el diseño como base, se realiza la implementación del pipeline y se muestran ejemplos de cómo las herramientas reaccionan ante diversos escenarios.

Abstract

Delivering software solutions guaranteeing product and development process security is an increasing necessity due to the surge of digital threats. The purpose of this work is to create a continuous delivery pipeline for Salesforce solutions using DevSecOps.

With this objective in mind, the life cycle phases that need to apply to Salesforce solutions are considered. For each phase, different tools are analyzed and compared to choose the one that best adjusts to the task. After the tools analysis, a design for the pipeline is proposed. Using this design as a basis, the pipeline is implemented and examples of how the tools react to various scenarios are shown.

Índice

Listas de figuras

Figuras del contenido

Figuras de los anexos

1. Introducción

1.1. Contexto y justificación del Trabajo

A lo largo de los últimos años, Salesforce ha realizado un esfuerzo por agilizar los procesos de despliegue de soluciones desde sus entornos de desarrollo a los entornos productivos. La creación de Salesforce DX [65] ha puesto a disposición de los desarrolladores una herramienta de línea de comandos que encapsula lo que antes era una mezcla de instrucciones API, herramientas declarativas y repetición de acciones manuales. Esto ha puesto al alcance de muchos el poder automatizar los procesos de despliegue al reducir sustancialmente la interacción y supervisión humana durante los despliegues.

Dada la naturaleza SaaS de Salesforce CRM [62], muchas de las empresas que trabajan con él dan por sentada la seguridad que la plataforma ofrece y no se preocupan de realizar análisis de seguridad durante el ciclo de desarrollo, confiando en que el sistema cubrirá las posibles vulnerabilidades. Sin embargo, aunque Salesforce ofrece garantías de protección a nivel de infraestructura e incluso código, no se hace responsable de las vulnerabilidades causadas por el desarrollo de código propio utilizando su lenguaje propietario Apex (backend) [22] o su modelo de componentes web LWC (frontend) [44].

En un escenario global de incremento de la ciberdelincuencia, en España durante el año 2022 hubo un incremento del 22,7% de delitos con respecto a 2021 [43], se hace necesario aplicar medidas de seguridad no solo sobre el producto final, si no también durante los procesos de desarrollo. En otras palabras, pasar del uso de DevOps a DevSecOps.

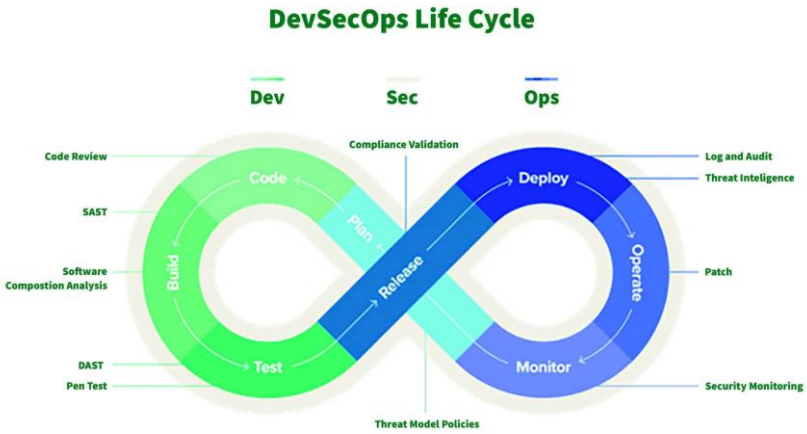


Figura 1: Diagrama DevSecOps
Fuente: DevSecOps Life Cycle, 2022, https://www.xalt.de/wp-content/uploads/2022/08/xalt_devsecops_cycle-scaled.jpg

Si DevOps agrupa los procesos y actividades de desarrollo y operaciones necesarios para llevar una solución o producto desde la fase de diseño hasta su consecución, el añadido de la capa de seguridad nos permite validar que el resultado de aplicar dichos procesos y actividades es un producto seguro de principio a fin del ciclo de desarrollo. OWASP [59] nos ofrece una serie pautas generales para aplicar la capa de seguridad, indicando posibles herramientas para cada fase. No todas las fases representadas aplican en el escenario de Salesforce. El software de Salesforce impide customizar las soluciones a nivel de infraestructura y tiene una política restrictiva con el uso de recursos de terceros [50][4], haciéndose responsable de la seguridad el proveedor en estos escenarios.

Este trabajo parte de la necesidad de implementar un ciclo de vida DevSecOps para desarrollos en Salesforce CRM, tanto del frontend como del backend, mediante un pipeline sobre el que actúen una serie de herramientas de seguridad según las necesidades de cada etapa sin recurrir a soluciones de terceros similares a Copado [16], dado que este tipo de herramientas suelen conllevar un alto coste económico en el que no se incluye el coste de las herramientas de seguridad. Para poder utilizar herramientas de seguridad hay que adquirirlas por separado e integrarlas, en algunos casos requiriendo desarrollo, en otros, el uso de conectores especializados.

1.2. Objetivos del Trabajo

El objetivo de este trabajo es desarrollar un pipeline automatizado de entrega continua que utilice las herramientas de seguridad adecuadas en cada fase del ciclo de desarrollo del software para la entrega de soluciones Salesforce. Para la consecución de este objetivo, se presentan los siguientes objetivos secundarios clasificados en tres tipologías:

1.2.1. Objetivos a nivel de investigación y estudios

- Revisión bibliográfica del estado del arte en seguridad de pipelines existentes en el contexto de Salesforce.
- Análisis de las distintas herramientas disponibles para reforzar la seguridad en dichos pipelines.
- Determinar que herramientas son necesarias para DevSecOps, que aportan cada una de ellas en este contexto y porque se han elegido unas u otras.

1.2.2. Objetivos a nivel de implantación y desarrollo

- Establecer un pipeline de entrega continua para desarrollos Salesforce.
- Configurar los diferentes sistemas de acuerdo con las prácticas de seguridad definidas.
- Realizar un proceso de entrega continua desde su inicio (desarrollo) hasta su final (despliegue en productivo) mostrando en cada paso como las prácticas de seguridad reaccionan a diferentes amenazas y los pasos necesarios para gestionar dichos escenarios.

1.2.3. Objetivos a nivel académico/entrega

- Cumplir con las entregas parciales definidas.
- Entregar la memoria final del Trabajo de Fin de Máster.
- Realizar la presentación del proyecto.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

1.3.1. Dimensión sostenibilidad

El impacto medioambiental de este trabajo viene definido por el impacto de las herramientas utilizadas para su desarrollo e implementación.

En el caso del desarrollo del trabajo, el impacto es mínimo, dado que el trabajo ha sido llevado a cabo desde un ordenador personal. No se ha incurrido en viajes ni desplazamientos que hayan generado huella de carbono.

En el caso de las herramientas analizadas y utilizadas en este trabajo, en cada uno de los apartados dedicados del capítulo 2 se puede ver donde consultar el impacto medioambiental para aquellas que ponen a disposición dicha información, así como una estimación sobre el producto final en el apartado 4.10.

1.3.2. Dimensión comportamiento ético y de responsabilidad social

La principal motivación tras la implementación de este trabajo desde una perspectiva ética es la necesidad de cumplir con los requerimientos legales de protección durante el proceso de desarrollo. En muchos casos, cumplir con los requerimientos de seguridad se contempla como una molestia en la que es mejor no ahondar, ya sea por el coste de horas (coste de recursos humanos) o el monetario (coste de las herramientas).

Como impacto negativo, la gente podría utilizar el conocimiento de la implementación para buscar vulnerabilidades y aprovecharse de ellas o realizar una implementación vulnerable a propósito.

Como impacto positivo, este trabajo permitirá una mayor agilidad a la hora de realizar la implementación de un pipeline seguro de desarrollo para soluciones Salesforce ya que elimina la barrera de entrada correspondiente a la falta de conocimiento.

1.3.3. Dimensión diversidad, género y derechos humanos

Este trabajo no tiene impacto en estas dimensiones, dado que se limita a los aspectos técnicos de la implementación de un pipeline. No hay un componente humano que pueda verse perjudicado o favorecido indebidamente.

1.4. Enfoque y método seguido

Para la definición de las etapas de seguridad a implementar, nos basamos en la definición de OWASP [59]. En el caso de Salesforce, al ser un SaaS, no entra en consideración las partes relacionadas con la seguridad de configuración y mantenimiento de la infraestructura. Además, Salesforce también cubre por nosotros la parte de SCA al aplicar restricciones en el uso de librerías de terceros para su frontend [50] y backend [4]. Por lo que nos quedarían las etapas que podemos ver en la Figura 2: **Etapas de seguridad para un pipeline de Salesforce**.

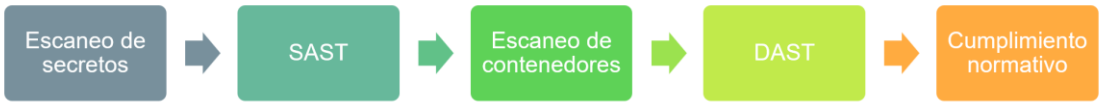


Figura 2: Etapas de seguridad para un pipeline de Salesforce
Fuente: elaboración propia

A alto nivel, la principal responsabilidad de cada etapa es:

Escaneo de secretos
Análisis del repositorio de código para detectar el uso no seguro de secretos.

SAST
Static Applicaction Security Testing. Análisis del código fuente.

Escaneo de contenedores
Análisis de las imágenes de contenedores Docker.

DAST
Dynamic Application Security Testing. Análisis de la aplicación a través de su frontal.

Cumplimiento normativo
Determinar se si cumple con las buenas prácticas y los procedimientos adoptados por la organización.

El proyecto tiene tres partes:

A) Investigación sobre las herramientas disponibles para cada etapa. Capacidades, limitaciones y procesos de implementación y configuración.

B) Análisis comparativo y selección de las herramientas a utilizar.

C) Implementación de la solución y ejecución de un proceso de desarrollo con ejemplos de cómo reaccionan las herramientas ante amenazas de seguridad.

El primer paso es realizar un análisis comparativo entre las diferentes herramientas que aplican en cada una de las fases de DevSecOps, que aporta cada una de ellas, sus requerimientos, costes, fortalezas y debilidades.

A partir de la información reunida mediante la investigación de las herramientas disponibles, procedemos a definir el diseño final. En esta parte del proyecto se justifica la selección de las herramientas y su configuración, así como los pasos a seguir para su implementación e interconexión.

Finalmente, la tercera parte del proyecto consiste en la implementación del diseño, la demostración de su uso y la actuación de las diversas herramientas sobre desarrollos con el objetivo de ponerlas a prueba y validarlas. Para ello se realizan una serie de despliegues de código con defectos para demostrar como el pipeline reacciona ante ellos. El código y los metadatos elegidos para la realización de los despliegues no pretenden conformar un aplicativo usable, si no meramente demostrar el funcionamiento de las herramientas de seguridad.

1.5. Planificación del Trabajo

1.5.1. Tabla de actividades

	Actividad	Inicio	Final	Días
1	**Planificación**	28/02/2024	12/03/2024	14
1.1	Definir alcance	28/02/2024	06/03/2024	8
1.2	Establecer objetivos	07/03/2024	08/03/2024	2
1.3	Definir metodología	09/03/2024	10/03/2024	2
1.4	Diagrama de Gantt	11/03/2024	11/03/2024	1
1.5	Redactar el plan de trabajo	11/03/2024	12/03/2024	2
1.6	Estado del arte	11/03/2024	12/03/2024	2
1.7	Entrega 01	12/03/2024	12/03/2024	1
2	**Análisis de herramientas**	13/03/2024	09/04/2024	28
2.1	Herramientas para repositorios Git	13/03/2024	15/03/2024	3
2.2	Herramientas de análisis de secretos	16/03/2024	17/03/2024	2
2.3	Herramientas SAST	18/03/2024	22/03/2024	5
2.4	Escaneo de contenedores	23/03/2024	24/03/2024	2
2.5	Herramientas DAST	25/03/2024	31/03/2024	7
2.6	Cumplimiento normativo	01/04/2024	06/04/2024	6
2.7	Entrega 02	07/04/2024	09/04/2024	3
3	**Diseño del pipeline**	10/04/2024	07/05/2024	28
3.1	DAFO Salesforce	10/04/2024	14/04/2024	5
3.2	Diseño de la solución	15/04/2024	28/04/2024	14
3.3	Justificación del diseño	29/04/2024	04/05/2024	6
3.4	Entrega 03	05/05/2024	07/05/2024	3
4	**Implementación del pipeline**	08/05/2024	11/06/2024	35
4.1	Configuración de las herramientas	08/05/2024	26/05/2024	19
4.2	Flujo completo de pipeline	27/05/2024	02/06/204	7
4.3	Entrega 04	03/06/2024	11/06/2024	9
5	**Presentación del TFM**	11/06/2024	24/06/2024	14

5.1	Presentación en vídeo	11/06/2024	18/06/2024	8
5.2	Preparación de la defensa	19/06/2024	23/06/2024	5
5.3	Defensa del TFM	24/06/2024	24/06/2024	1

1.5.2. Diagrama de Gantt

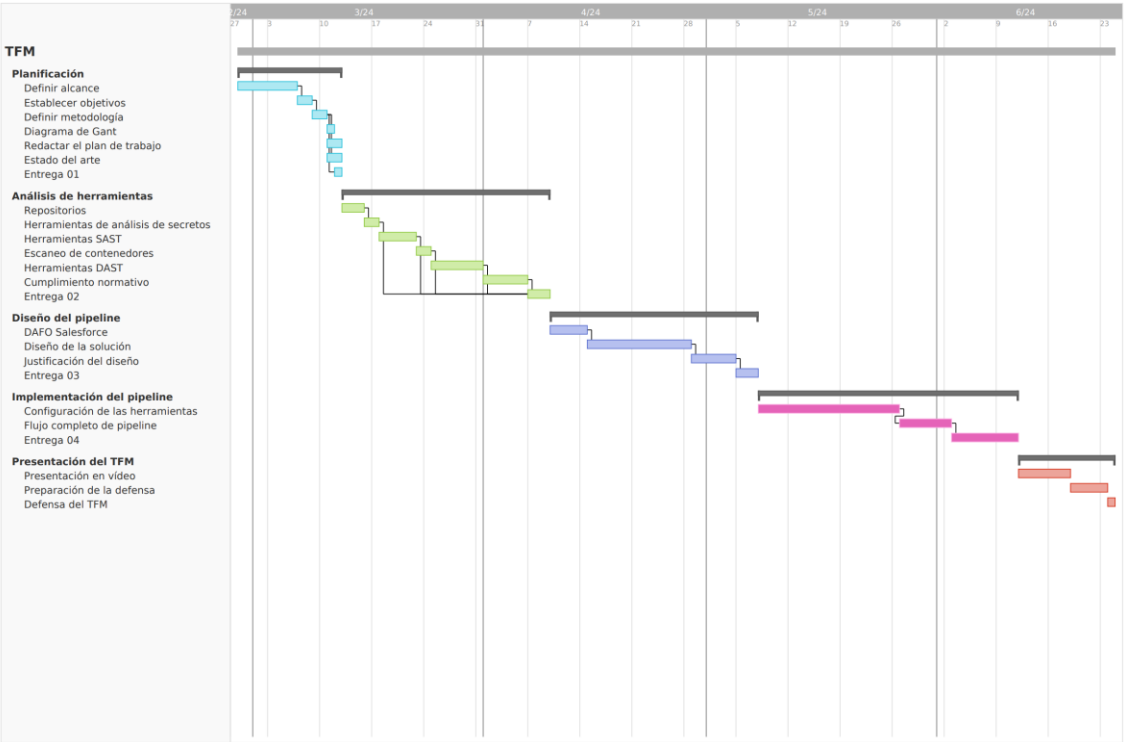


Figura 3: Diagrama de Gantt
Fuente: elaboración propia

1.6. Estado del arte

1.6.1. Herramientas DevOps para Salesforce

Tomando como referencia el artículo publicado en 2021 sobre la construcción de pipelines de entrega continua [42] por Salesforce Ben, una de las webs de referencia sobre noticias e información relacionadas con Salesforce, podemos observar tres grandes tipologías de implementación para DevOps:

1. Soluciones creadas por terceros con mayor o menor cohesión con el software de Salesforce que ofrecen una simplificación en la gestión de herramientas de DevOps y DevSecOps. Dichas herramientas han de obtenerse y pagar por separado. Ejemplos de este tipo de soluciones son:
    a. Copado [16], que está en gran parte desarrollado sobre la propia plataforma de Salesforce como un paquete gestionado validado por la AppExchange [4].

b. Autorabit [6], externo a la plataforma de Salesforce como servicio cloud que además de gestión del ciclo de vida del desarrollo, ofrece otros servicios como copias de seguridad y restauraciones en caso de pérdidas de datos.

Hasta hace unos años, la principal motivación en el uso de estas soluciones radicaba en la gestión de los despliegues de paquetes de metadatos entre los entornos de desarrollo y producción de Salesforce que en algunos casos puede resultar un proceso complejo. La llegada de Salesforce DX [65] ha agilizado este proceso al unificar las metodologías existentes de despliegue.

2. Soluciones generalistas, creadas por terceros sin relación directa con Salesforce que permiten gestionar múltiples herramientas de forma comprensiva. Ejemplos son:
    a. Jenkins [47] es un servidor opensource enfocado a la automatización de procesos.
    b. Bamboo [7] permite la implementación de pipelines de desarrollo. Al ser propiedad de Atlassian, se integra con facilidad con otros de sus productos como el repositorio de código Bitbucket [8] o la herramienta de gestión de proyectos Jira [48].

3. Uso de las capacidades de integración continua de plataformas de control de versiones:
    a. Bitbucket Pipelines [14] como una alternativa más simple que Bamboo.
    b. GitHub [37] tiene a su disposición GitHub Actions.

1.6.2. Herramientas DevSecOps para Salesforce

La plataforma de Salesforce tiene la capacidad nativa para realizar test de código unitarios sobre código Apex [84]. Para poder realizar despliegues en productivo de código, se requiere ejecutar test que recorran por lo menos el 75% de las líneas de código ejecutable existentes sin producirse errores. Los test permiten realizar pruebas de visibilidad de datos además de validar funcionalidades. Sin embargo, por lo general, los test se crean para cumplir con el requisito del 75% no para hacer validaciones de seguridad en el acceso de datos o de las funcionalidades. Los test pueden ejecutarse de forma selectiva mediante Salesforce DX y recuperar los resultados fuera del sistema. No existe capacidad nativa de test para los LWC, otros elementos de configuración de la UI o herramientas declarativas.

Las herramientas de análisis para superar la revisión de seguridad de la AppExchange están enfocadas al análisis de paquetes y requieren de partnership con Salesforce para ser utilizadas. No es parte de este trabajo abordar un pipeline para el desarrollo de paquetes para la AppExchange.

Con respecto a las herramientas para las etapas definidas en el apartado 1.4 procederemos a analizarlas y compararlas en el capítulo 2.

1.7. Breve sumario de productos obtenidos

El producto generado por este trabajo en un pipeline CI/CD para soluciones Salesforce con detección de secretos, análisis de calidad de código, despliegue automatizado y análisis web sobre el que se han aplicado una serie de comprobaciones de seguridad a nivel de desarrollo y de gestión de las diversas herramientas utilizadas.

El pipeline está compuesto por:

- Repositorio Git utilizando GitHub para contener el código del proyecto y gestionar los diversos automatismos implementados, así como los accesos de los diferentes usuarios y medidas de protección del repositorio.
- GitHub Action de GitLeaks.
- Configuración de un entorno de SonarCloud vinculado al repositorio de GitHub para realizar análisis SAST del código.
- Configuración de SonarLint sobre Visual Studio Code para analizar el código con las reglas de SonarCloud antes de su subida al repositorio.
- DockerHub con DockerScout para almacenar y analizar las imágenes Docker necesarias para automatizar los despliegues a los entornos de Salesforce.
- GitHub Action para lanzar la imagen de Docker que realiza los despliegues.
- Uso de ZAP para análisis web.

1.8. Breve descripción de los otros capítulos de la memoria

- Capítulo 2: Análisis de herramientas
    - Para cada fase del pipeline selecciono dos herramientas que pueden cubrir las necesidades asociadas, las analizo y las comparo entre ellas.

- Capítulo 3: Diseño del pipeline
    - Elijo que herramienta utilizar para cada una de las fases del pipeline, justifico porqué y explico como las voy a querer utilizar, así como los pasos que ejecutaremos en el pipeline.

- Capítulo 4: Implementación del
    - Descripción de como he implementado la solución, configuración de las diferentes herramientas, ajustes sobre el diseño, dificultades encontradas y soluciones.

- Capítulo 5: Conclusiones y trabajos futuros
    - Indico las conclusiones del trabajo, los puntos pendientes, las mejoras y siguientes pasos a seguir.

- Capítulo 6: Glosario
    - Glosario de términos utilizados en el trabajo por orden alfabético.

- Capítulo 7: Bibliografía
    - Listado de recursos consultados y referenciados en el trabajo por orden alfabético.

- Anexos
    - Contiene las guías para dar de alta las cuentas necesarias para utilizar las diferentes herramientas, el proceso de subida de código inicial, la lista de programas utilizados en este trabajo y una descripción de los ficheros adjuntos a este trabajo.

2. Análisis de herramientas

En este apartado voy a analizar diferentes herramientas disponibles para cada una de las cinco etapas indicadas en apartado 1.4, así como herramientas de control de repositorio. Para cada apartado aporto una breve introducción sobre los principales objetivos a cubrir con las herramientas y su razón de ser, seguida de una descripción de sus capacidades con respecto a la auditoría, control de acceso, división de responsabilidades y huella de carbono.

Algunas de las herramientas contempladas pueden aplicarse en más de una etapa o tienen funcionalidades que se solapan. En el capítulo 3, indico que funcionalidades de cada herramienta se utilizan o descartan para la implementación propuesta y razono los motivos que me llevan a ello.

Huella de carbono

La mayoría de los proveedores de soluciones SaaS no informan del impacto ambiental detallado del uso de sus sistemas [30]. Las principales razones que se dan por ello son:

- La dificultad de aislar el coste de una operación específica, dado que la energía requerida puede variar según la congestión de la red, la máquina/sistema que ejecuta la operación según la disponibilidad, y el origen de la energía en cuestión que puede proceder de fuentes renovables o de combustibles fósiles dependiendo del estado del mercado.
- El impacto de la construcción, mantenimiento y destrucción de las estructuras físicas utilizadas.
- La distribución de las responsabilidades. Los proveedores utilizan múltiples plataformas y servicios sobre los cuales, en algunos casos, no tienen control.

Green Software Foundation [41] es un proyecto que realiza un seguimiento del consumo de energía y el impacto ambiental de los servicios Cloud por proveedor, pero no aporta la granularidad necesaria para un análisis detallado.

En general, las herramientas que comento en este apartado se pueden clasificar en tres tipologías:

- Herramientas SaaS con objetivos de reducción de la huella de carbono.
- Herramientas ejecutables sobre las herramientas anteriores.
- Herramientas que no ofrecen información sobre su impacto.

Para este apartado me centraré en indicar las metas medioambientales de las compañías que las tienen. Para el análisis de impacto específico a la implementación, ver apartado 4.10.

2.1. Herramientas para repositorios Git

Partiendo de la base de que se utilizará un repositorio de Git sobre el cual realizar el control de versiones del código y metadatos de la solución, necesitaremos una forma de agilizar y automatizar los procesos de gestión. Existen muchas herramientas que ofrecen funcionalidades en este sentido con diversos niveles de configuración y personalización.

En esta sección me centro en las soluciones de GitHub [37] y Bitbucket [8], dado que son las que utiliza la empresa en la que estoy empleado actualmente. Ambas soluciones nos permiten gestionar la configuración e interacción con el repositorio Git mediante una interfaz de usuario. Además, ofrecen posibilidades de automatización e integración con herramientas de terceros.

2.1.1. GitHub

Sistema de control de versiones Git con opción de servicio Cloud y on-premise. Para este trabajo en concreto, contemplaré la vertiente Cloud de la herramienta.

Audit Log y control de acceso

GitHub tiene trazabilidad de accesos y acciones de sus usuarios. Los logs de las cuentas Enterprise pueden consultarse a través de una API Rest y la API de GraphQL. Para el resto de las cuentas de GitHub, el acceso está limitado a la interfaz de usuario [21].

GitHub permite SSO de terceros. Sin embargo, el funcionamiento es ligeramente diferente al resto de sistemas que implementan SSO, en tanto que las cuentas pueden estar asociadas a diferentes sistemas SSO. El sistema al que se solicitará el acceso dependerá de la cuenta a la que se esté intentado acceder.

Separación de responsabilidades

GitHub permite la configuración de roles (conjuntos de permisos) que aplican a usuarios o grupos de usuarios. Los roles difieren según las tipologías de organización, pero para todas ellas existe distinción entre los usuarios del repositorio y sus administradores [60].

Automatización de procesos

Para la automatización de procesos, GitHub nos ofrece GitHub Actions [28]. Mediante esta herramienta se pueden configurar flujos de trabajo que realicen acciones sobre el repositorio a partir de los eventos que tengan lugar en este.

Para la ejecución de estas acciones, GitHub instancia una máquina virtual en la que ejecutarlas [1]. Las máquinas virtuales pueden utilizar entornos Ubuntu, Windows y MacOS. Como alternativa, se puede lanzar un contenedor Docker directamente.

Huella de carbono

Con respecto a la ejecución de software, GitHub es neutral en generación de carbono desde 2019, tiene como objetivo utilizar únicamente fuentes de energía renovable para el 2025 y alcanzar un balance de generación de carbono negativo para el 2030 [32].

Con respecto al mantenimiento, construcción y destrucción de infraestructuras físicas, GitHub tiene como meta reducir la generación de desperdicios. Así mismo, tiene como objetivo alcanzar un consumo negativo de agua para 2030 [32].

2.1.2. Bitbucket

Sistema de control de versiones Git con opción de servicio Cloud y on-premise. Para este trabajo en concreto, contemplaré la vertiente Cloud de la herramienta.

Audit Log y control de acceso

Bitbucket tiene trazabilidad de accesos y acciones de sus usuarios. Los logs son accesibles mediante API y a través de la interfaz de usuario [86].

Bitbucket permite SSO mediante Atlassian y herramientas de terceros.

Separación de responsabilidades

La administración del sistema puede separarse del uso del repositorio como tal. Bitbucket permite asignar permisos a nivel de repositorio por usuario, pero también por grupos de usuarios, lo que facilita identificar los permisos asignados y su modificación [58].

A nivel de repositorio, se puede configurar el acceso de los usuarios a las diferentes ramas por tipologías. Un usuario puede hacer commits contra las ramas de desarrollo sin impedimentos, pero requerir de autorización para poder manipular otras ramas como la main.

Automatización de procesos

Para la automatización de procesos, Bitbucket ofrece Bitbucket Pipelines [14]. Mediante esta característica de la plataforma, podemos ejecutar acciones en función de condiciones específicas como sería un commit contra una rama determinada. La complejidad de las instrucciones a llevar a cabo puede variar desde la ejecución de un script de código a lanzar un Docker. También permite integraciones con otros sistemas.

Atlassian ofrece, además, una herramienta específica de automatización no exclusiva de Bitbucket, Bamboo [7]. Esta herramienta tiene una mayor facilidad y control a la hora de integrar diferentes herramientas, así como características de base. Sin embargo, la gestión de usuarios se realiza por separado, requiere de integración con Bitbucket y su mayor complejidad puede no ser excesiva para algunos proyectos.

Huella de carbono

La huella de carbono de Bitbucket y Bamboo queda bajo el paraguas de las políticas de Atlassian. Los informes anuales sobre el impacto medioambiental y las medidas aplicadas para reducirlo se pude consultar en [5]. Atlassian tiene como objetivo alcanzar la neutralidad en generación de emisiones de carbono para el 2040.

2.2. Herramientas de protección de secretos

Las herramientas de protección de secretos se encargan de restringir el acceso y la modificación de información privada que, por motivos de seguridad, no puede ser accesible de forma universal.

2.2.1. Git-secret

Git-secret [35] es una herramienta que permite el cifrado de ficheros mediante un sistema de claves pública/privada (RSA) que funciona sobre los comandos de Git. Permite mantener el control de versiones sobre los ficheros cifrados de modo que las configuraciones de los secretos se alinean con las del estado del repositorio en un momento dado.

Audit Log y control de acceso

El alta de los usuarios se realiza por comandos. No existe control de acceso.

Los accesos a secretos se definen a nivel de usuario, pero no hay un sistema de Audit de las peticiones de desencriptación. La responsabilidad de la trazabilidad de cambios en los secretos queda en manos del repositorio Git (en esencia, el usuario asociado al commit que ha realizado el cambio). No hay trazabilidad para el alta o baja de usuarios.

Dada la simplicidad de la herramienta, no permite integración directa con sistemas de SSO.

Separación de responsabilidades

Carece de sistema de roles. Los accesos se definen a nivel de usuario por secreto.

Huella de carbono

Es un programa instalable por lo que su impacto irá en función de la máquina que lo ejecute y, por tanto, deberá tenerse en consideración durante su implementación.

2.2.2. AWS Secret Manager

AWS Secret Manager [9] es un producto de AWS que permite almacenar secretos en la nube de forma segura mediante cifrado. Es una solución SaaS que permite la consulta de secretos a través de interfaz y API para aquellas aplicaciones de terceros que necesiten usarlos.

Audit Log y control de acceso

Los accesos a los secretos mediante interfaz y API quedan registrados en el sistema y pueden recuperarse a través de la API de Audit Logs.

Permite integración con sistemas de SSO propios de AWS o de terceros.

Separación de responsabilidades

Separa acceso y modificación de secretos. No hay sistema de roles propio de la herramienta, aunque sí se pueden utilizar configuraciones más avanzadas si se utiliza en conjunto con otros productos de AWS.

Huella de carbono

Amazon Web Services tiene el compromiso de usar solo energía renovable para el 2025, aunque para algunas regiones específicas este objetivo ya está cumplido. Para el 2040 espera alcanzar la neutralidad en emisiones de carbono [2].

Dispone de una herramienta para analizar el impacto ambiental de sus productos a nivel de cliente.

2.3. Herramientas de detección de secretos

Las herramientas de detección de secretos son consideradas en muchos casos como herramientas SAST, pero dado lo específico de su tarea y siguiendo la guía de OWASP, las trato como un punto a parte.

Estas herramientas se especializan en encontrar el uso de información privada en abierto o de forma que contravenga las medidas de seguridad requeridas como sería escribir contraseñas en texto plano dentro de ficheros y alertar de ello.

2.3.1. GitLeaks

GitLeaks es una herramienta opensource que permite el análisis de repositorios Git [39] es compatible con GitHub Actions y puede ser ejecutada en imágenes Docker. Permite analizar los ficheros mediante un conjunto de reglas preestablecidas y a definir por los usuarios, para encontrar patrones que sugieran el uso de contraseñas en abierto.

Audit Log, control de acceso y separación de responsabilidades

La herramienta es parte de los sistemas que la ejecutan, por lo que estas responsabilidades quedan en mano del sistema huésped.

Huella de carbono
Al ser una herramienta de comandos, la huella de carbono generada por su uso será parte del sistema que la ejecute.

2.3.2. GitGuardian

GitGuardian [36] es una solución SaaS dedicada a la detección de secretos en repositorios Git capaz de integrarse directamente con GitHub y Bitbucket entre otros.

La herramienta dispone de una serie de parámetros de búsqueda predefinidos y permite configurar reglas propias. Pueden aplicase un sistema de clasificación de incidencias en función de su importancia y establecer alertas automatizadas.

Tiene capacidad para analizar imágenes Docker en busca de secretos desprotegidos o vulnerables.

Audit Log y control de acceso

Permite integración con sistemas de SSO de terceros. Mantiene un Log de accesos a la plataforma por usuario y las acciones que estos realizan. Dispone de una API para extraer los Audit Logs.

Separación de responsabilidades

Las responsabilidades de gestión de la plataforma y su uso están separadas. Se pueden crear y gestionar equipos de usuarios con diferentes permisos asignados.

Huella de carbono

A fecha de la publicación de este trabajo, no hay información sobre el impacto ambiental de GitGuardian en su web, blog o FAQ.

2.4. Herramientas SAST

Las herramientas de análisis de código estático [79] también conocidas como herramientas de análisis de código nos ayudan a detectar posibles vulnerabilidades de seguridad durante las primeras etapas de desarrollo del código agilizando su resolución. Es importante la detección temprana de estos problemas para evitar incidencias en la aplicación final y el coste que representa desarrollar su solución, que puede conllevar modificaciones complejas de la lógica establecida.

Las herramientas SAST trabajan sobre el código sin compilar y pueden ofrecer información y consejos de como solventar las vulnerabilidades o errores detectados. Las más complejas permiten informar de falsos positivos para ignorarlos en análisis posteriores.

Los análisis pueden realizarse en vivo, mientras el desarrollador escribe el código en su IDE o procesador de textos ya sea mediante extensiones o como parte integra del sistema utilizado, o de forma periódica sobre el repositorio del código.

2.4.1. SonarCloud/SonarLint

SonarCloud [70] es una solución SaaS que permite el análisis de bases de código en busca de vulnerabilidades de seguridad, uso incorrecto de expresiones, problemas de mantenibilidad, escalabilidad, detección de secretos y seguimiento de buenas prácticas de programación [71].

Tiene capacidad de integración de base sobre GitHub y Bitbucket, permitiendo trabajar con la herramienta a través de la interfaz de los repositorios. Con SonarLint [75] podemos instalar un plugin en Visual Studio Code para el análisis en vivo del código que extrae los patrones a seguir de SonarCloud.

SonarCloud soporta múltiples lenguajes, entre los que se encuentran los de nuestro interés: JavaScript, CSS y Apex. Para cada lenguaje existen diferentes configuraciones de análisis por defecto a las que SonarCloud denomina Perfiles. La configuración para utilizar dependerá de las funcionalidades que queramos cubrir con el desarrollo del código. No es lo mismo usar JavaScript para el desarrollo web que para montar un servidor Node.js.

Los Perfiles se pueden crear de cero, en cuyo caso la definición de los patrones de búsqueda queda a cargo del cliente, o pueden utilizarse los ya existentes como base sobre la que añadir nuevas reglas. Para ello tenemos dos aproximaciones:

- Clonación de Perfiles

  Con la clonación, conservamos la configuración base del Perfil original en el momento de la copia. La responsabilidad de mantener actualizado este Perfil recae en el cliente.

- Extensión de Perfiles

  Con la extensión, mantenemos el nuevo Perfil alienado con el original. Cualquier cambio en el original se verá reflejado de forma automática. La desventaja de ello es que perdemos el control de cuando entran en funcionamiento nuevas reglas, pero no hay que preocuparse por mantener al día el Perfil de forma manual.

Los Profiles se usan para definir las Quality Gates de SonarCloud. Este es un aspecto no disponible en SonarLint. Las Quality Gates son un conjunto de medidas de calidad a cumplir durante los análisis de código como puede ser mantener la complejidad de las funciones o el número de posibles vulnerabilidades por debajo de un umbral determinado. Se pueden tener múltiples Quality Gates dependiendo de cuando se esté ejecutando el análisis del código.

SonarCloud sigue la filosofía de Clean As You Code. La idea principal de esta filosofía es centrarse en mantener el código nuevo limpio e ir mejorando progresivamente el código existente para evitar tener que realizar restructuraciones masivas del código. Para ello, permite diferenciar entre código viejo y nuevo en función de una serie de parámetros, como puede ser la fecha de última modificación, y aplicar Quality Gates especificas en cada caso.

Audit Log y control de acceso

A fecha de la publicación de este trabajo, SonarCloud carece de Audit Log a nivel de administración del sistema, tampoco para el registro de incidencias. Queda en manos de los repositorios el indicar el responsable del commit de la incidencia.

El acceso a las características de análisis se realiza desde los repositorios a los cuales SonarCloud está integrado, por lo que no es necesario dar de alta a los desarrolladores en el sistema.

No tiene capacidad de SSO.

Separación de responsabilidades

Sonar Cloud permite la asignación de permisos a nivel de usuario y proyecto. La administración de la cuenta está separada de la administración de los proyectos [38]. Los permisos para ignorar Quality Gates o bloqueos quedan en manos de los repositorios integrados, dado que son ellos los que establecen el comportamiento en función del resultado de las Quality Gates. SonarCloud se limita únicamente al análisis.

Huella de carbono

A fecha de la publicación de este trabajo, no hay información sobre el impacto ambiental de SonarCloud en su web, blog o FAQ.

El impacto de SonarLint quedará reflejado sobre las máquinas que lo ejecuten.

2.4.2. Semgrep

Semgrep es una solución SaaS que permite el análisis de bases de código en busca de vulnerabilidades de seguridad, uso incorrecto de expresiones, problemas de mantenibilidad y seguimiento de buenas prácticas de programación [68]. La detección de secretos está gestionada por un producto separado y dedicado.

Tiene capacidad de integración de base sobre GitHub y Bitbucket, permitiendo trabajar con la herramienta a través de la interfaz de los repositorios. Existe una extensión para Visual Studio Code para el análisis del código cada vez que se abre o modifica un fichero [69] y una CLI instalable que permite su ejecución de forma manual o automatizada.

Semgrep tiene múltiples lenguajes de programación disponibles, pero no tiene soporte para CSS y Apex está en fase Beta, por lo que las reglas necesarias para dichos lenguajes deberían de implementarse desde cero. Semgrep tiene un registro de reglas colaborativo a partir del cual hay que seleccionar las que queremos aplicar. Permite la creación de reglas propias.

Para las ejecuciones de Semgrep desde la plataforma SaaS, Semgrep generará registros denominados findings por cada incidencia que encuentre. Tiene la capacidad de informar de falsos positivos.

Audit Log y control de acceso

A fecha de la publicación de este trabajo, Semgrep carece de Audit Log a nivel de administración del sistema, tampoco para el registro de incidencias. Queda en manos de los repositorios el indicar el responsable del commit de la incidencia.

Los desarrolladores pueden consultar las incidencias detectadas desde el repositorio de código o desde la aplicación web.

Semgrep permite configurar SSO.

Separación de responsabilidades

Semgrep diferencia entre administradores y miembros, separando la gestión de la plataforma de su uso. En función del rol asignado, se pueden restringir el uso de las características de la plataforma. Hay cierto riesgo de seguridad, dado que el rol por defecto que se asigna durante la creación de usuario es el de administrador [41].

Huella de carbono

A fecha de la publicación de este trabajo, no hay información sobre el impacto ambiental de Semgrep en su web, blog o FAQ.

El impacto de las extensiones y la CLI quedará reflejado sobre las máquinas que lo ejecuten.

2.5. Escaneo de contenedores

El pipeline que este trabajo propone no será utilizado para controlar el ciclo de desarrollo de imágenes Docker como tarea principal, pero hemos de contemplar su análisis de seguridad debido a que utilizaremos imágenes Docker para el control de los despliegues entre entornos de Salesforce y para ejecutar algunas de las herramientas de seguridad que no seamos capaces de integrar directamente sobre los automatismos del repositorio elegido.

OWASP nos ofrece una lista de los principales consejos, buenas prácticas y peligros para tener en cuenta a la hora generar imágenes Docker seguras [27]. La regla número nueve nos habla, precisamente, de la integración de herramientas de seguridad en un pipeline de desarrollo continuo. Algunas de las herramientas sugeridas son de propósito general, con capacidad de análisis no solo para imágenes Docker. Para este apartado, contemplaré Docker Scout [26] y Docker Bench [24] dado que son herramientas dedicadas.

2.5.1. Docker Scout

Docker Scout es una herramienta dedicada para el análisis de imágenes Docker ofrecida por la propia compañía de Docker. El análisis se realiza sobre las imágenes ya compiladas y almacenadas en un repositorio. Por defecto, Docker Scout siempre tomará la última versión disponible de la imagen y ofrecerá una lista de vulnerabilidades detectadas, así como de posibles soluciones para estas.

Adicionalmente, podemos utilizar la herramienta para definir una serie de políticas a evaluar cómo sería el tipo de licencias utilizadas por los elementos de la imagen, la existencia de elementos caducos o particularidades de la configuración que no queremos que tengan lugar.

Docker Scout ofrece integración con GitHub y una herramienta de visualización de resultados en formato Dashboard para repositorios conectados.

2.5.2. Docker Bench

Docker Bench utiliza como base la versión 1.6.0 de CIS Docker Benchmark, un conjunto de normas y buenas prácticas propuestas y mantenidas por la comunidad de Center for Internet Security [11].

La herramienta se puede ejecutar mediante comando sh o como un contenedor Docker. Se recomienda la construcción de la imagen Docker a partir de la última versión disponible de Docker Bench dado que la imagen disponible en Docker está desactualizada.

La ejecución de la herramienta genera una lista de resultados con una entrada por cada evaluación indicada en el CIS Docker Benchmark. Si bien se puede regular la cantidad de elementos a evaluar, no podemos añadir nuevas reglas por nuestra cuenta utilizando las instrucciones disponibles. En caso de utilizar la imagen Docker, los resultados se almacenarán en dos archivos dentro del contenedor.

2.6. Herramientas DAST

Las herramientas de análisis de vulnerabilidades [88] también conocidas como herramientas de análisis de seguridad dinámico de aplicaciones nos ayudan a detectar posibles vulnerabilidades en tiempo de ejecución. Los análisis se realizan operando sobre el frontend de la aplicación y sus servicios expuestos mediante API. Para ello, se aplican diferentes tipologías de ataque y manipulación y se compara el comportamiento esperado del aplicativo con el obtenido.

2.6.1. Invicti (Netsparker y Acunetix)

Invicti [45] nace de la adquisición de Netsparker y Acunetix (entre otras). Ofrece una solución de análisis DAST como servicio SaaS u on-premise. Dependiendo del plan, dispone de diferentes niveles y tipos de análisis y características. Tiene capacidad para realizar análisis de interfaz de usuario, API, composición del software, pruebas de autenticación y vulnerabilidades conocidas. Permite programar los análisis e interaccionar con la herramienta a través de una API Rest para facilitar la automatización de la ejecución de los análisis y su interpretación, además de tener integración con GitHub y Bamboo.

Existen integraciones adicionales con otros sistemas de gestión de proyectos como Jira, herramientas de comunicación para equipos como Stack y gestión de secretos entre otras.

Hay que tener en cuenta que la utilidad de estas integraciones depende de que tipos de análisis se ejecuten y que características de la solución se quieran aplicar, dado que Invicti puede utilizarse también para realizar análisis de infraestructura IAST y de composición de software SCA. Este último limitado a aplicaciones PHP, Node.js, Java

y .NET sin posibilidad de adaptar con reglas propias las configuraciones existentes para otros lenguajes o entornos.

Existe una alta complejidad de configuración y uso dado que la solución es, en esencia, la suma de múltiples herramientas individuales gestionadas de forma transparente a través de una aplicación única.

Audit Log y control de acceso

No dispone de Audit Log para los accesos de administración. Permite SSO con sistemas de terceros.

Separación de responsabilidades

Tiene un sistema de roles para la parte de administración y visualización de resultados de los análisis mediante diagramas e interfaz de usuario.

Huella de carbono

A fecha de la publicación de este trabajo, no hay información sobre el impacto ambiental de Invicti en su web, blog o FAQ. Para la solución on-premise, el impacto dependerá de las máquinas donde se ejecute.

2.6.2. Zed Attack Proxy (ZAP)

Zed Attack Proxy [91] es una solución sin ánimo de lucro soportada por Crash Override Open Source Fellowship que utiliza las directrices de OWASP para el análisis de aplicaciones. Se trata de un Proxy que hace de man-in-the-middle entre las comunicaciones del frontend y el backend de la aplicación web a analizar.

ZAP soporta análisis pasivos y activos tanto de forma manual como automatizada. Se pueden utilizar filtros de alertas para descartar los falsos positivos y se pueden añadir falsos negativos mediante desarrollo de scripts de detección. ZAP recomienda notificar de este tipo de escenarios para poder incluirlos en la base del programa.

La solución está disponible como instalable para sistemas operativos Windows, Linux y MacOS, aunque en este último no está reconocido como desarrollador y hay que dar permisos para poder realizar la instalación. También se puede ejecutar como Docker y tiene compatibilidad con GitHub Actions además de API y modo Daemon.

Las capacidades de la solución pueden ampliarse mediante el uso de add-ons, que pueden obtenerse a través de un marketplace [90]. En caso de que los add-ons existentes no cumplan con lo requerido, se pueden desarrollar add-ons propios.

Existen algunas soluciones más completas que trabajan utilizando ZAP de base como SOOS [78] cubriendo algunas de sus carencias y expandiendo funcionalidades sin necesidad de tener que gestionar con tanto detalle la herramienta.

Audit Log, control de acceso y separación responsabilidades

ZAP no ofrece gestión de equipos o usuarios de base, por lo que no hay Audit Logs de sus actividades. Podría utilizarse un add-on para registrar las actividades del uso de la herramienta y dejar la identificación del usuario que la utiliza en manos del sistema huésped.

Huella de carbono

Al ser un programa instalable, el impacto ambiental se reflejará en las máquinas que ejecuten el programa.

2.7. Cumplimiento normativo

Como solución SaaS, Salesforce mantiene una larga lista de certificaciones de seguridad [64]. Dada su naturaleza, no hay una herramienta de análisis para validar el cumplimiento de la herramienta en sí. Existen niveles de cifrado y seguridad diferenciados para aplicaciones que funcionan sobre la plataforma que gestionan datos sensibles como sería el caso de datos médicos.

Por ello, voy a centrar este apartado en la auditoría del uso de las herramientas desplegadas sobre el pipeline de desarrollo continuo con el objetivo de centraliza la detección de usos indebidos de las mismas. Mediante el análisis de los diferentes Audit Logs podemos determinar:

- Comportamientos anómalos de los usuarios.
- Modificaciones no autorizadas de las configuraciones.
- Accesos irregulares a las herramientas.
- Altas/Bajas o modificaciones de responsabilidades de usuarios.

La mayoría de las herramientas mencionadas en los apartados anteriores ofrecen la capacidad de registrar Audit Logs. Las herramientas que considero en este apartado no son específicas de cumplimiento normativo, si no de unificación y gestión de logs del funcionamiento de programas, pero pueden adaptarse para el escenario planteado.

2.7.1. Splunk

Splunk [80] es un servicio SaaS u on-premise que permite recopilar datos desde diferentes sistemas para analizarlos y generar visualizaciones comprensivas mediante gráficos e informes. Los datos pueden obtenerse mediante integraciones o cargas de ficheros.

Ofrece diversos productos focalizados en la detección de amenazas y comportamientos de usuarios [82]. Permite la configuración de KPIs y de alertas sobre

estos. Las alertas pueden enviarse por email o a través APIs de terceros como Slack, Jira o AWS.

Los datos almacenados por Splunk son encriptados con AES 256-bit por defecto y permite la gestión de claves de encriptado propias. Dispone de certificaciones para el tratamiento de datos médicos (HIPAA) entre otras.

Audit Log y control de acceso

Splunk permite el uso de SSO de terceros y dispone de un sistema de Audit Logs sobre la propia plataforma.

Separación de responsabilidades

Los roles de administración y uso de la plataforma están separados y los permisos pueden establecerse a nivel de usuario mediante la asignación de roles. Tiene un sistema de equipos para facilitar la agrupación y gestión de los usuarios.

Huella de carbono

Splunk se compromete a reducir sus emisiones de carbono para llegar a ser neutral de cara al 2050 con metas intermedias para el 2030 y 2040 [81]. Actualmente, la infraestructura y servicios de terceros utilizados por Splunk utilizan únicamente energía renovable.

Ofrecen apoyo y guía para aquellas compañías que quieran implementar políticas de control de emisiones de carbono mediante el análisis y monitorización de los procesos necesarios para alcanzar la meta de neutralidad.

2.7.2. OpenSearch

OpenSearch [53] es una solución on-premise para el análisis, procesamiento y visualización de grandes volúmenes de datos. Está disponible como imagen para contenedores Docker. Subdivide sus funcionalidades en tres módulos que pueden customizarse libremente:

- **OpenSearch**
  El motor de búsqueda, encargado de indexar y realizar las consultas contra los datos disponibles.

- **OpenSearch Dashboards**
  El generador de informes y gráficos a partir de los datos disponibles.

- **Data Prepper**
  El servidor que se encarga de la recolección y procesado de los datos para su almacenado.

Los datos almacenados en el sistema pueden enmascararse y ofuscarse además de restringir el acceso a usuarios con roles específicos. La configuración del sistema puede realizarse a través de una API Rest.

Audit Log y control de acceso

Existe un plugin para la gestión de la seguridad de usuarios y roles. Se puede utilizar un sistema de autenticación externo o Active Directory para gestionar los usuarios y sus permisos. Tiene opciones para establecer SSO y un sistema de Audit Logs para el control de las acciones de los usuarios sobre la plataforma.

Separación de responsabilidades

La herramienta tiene una gran flexibilidad a la hora de definir roles y permisos. Por defecto, tiene roles separados para las gestiones de administración y uso.

Huella de carbono

Al ser una solución on-premise, el impacto ambiental se reflejará en las máquinas que ejecuten el programa.

3. Diseño del pipeline

En este capítulo indico y justifico las herramientas que la solución utiliza, sus responsabilidades y funcionalidades. Así mismo, indico el flujo de uso tanto para los usuarios finales, los miembros del equipo que utilizarán el pipeline para iterar y entregar soluciones, como para los administradores del conjunto del pipeline.

3.1. Organización de los usuarios

Tendremos dos equipos con los siguientes roles:

Equipo de usuarios finales

a. Desarrolladores
    i. Su responsabilidad radica en el desarrollo del código y configuración de la solución Salesforce a entregar.
    ii. Crearán ramas de desarrollo en el repositorio y las actualizará con sus modificaciones.
    iii. Solicitará la promoción de cambios para procesos de QA.

b. QA
    i. Validarán las peticiones de promoción de los desarrolladores para su análisis, prueba y validación.

c. Líder técnico y funcional
    i. Dará el visto bueno para la entrega final de las soluciones Salesforce.

Equipo de administradores

a. Administrador de Sistema
    i. Será el responsable del alta y baja de administradores en los diferentes sistemas.
    ii. Tendrá la propiedad de las distintas cuentas y gestionará los pagos necesarios.
    iii. Solo habrá un Administrador de Sistema por sistema.

b. Administradores
    i. Serán los responsables de configurar las herramientas y dar de alta a los usuarios finales.
    ii. Monitorizarán el uso de las herramientas para asegurar su correcto funcionamiento.
    iii. Actualizarán las herramientas y sus funcionalidades según sea necesario para su correcto funcionamiento.

3.2. Salesforce

3.2.1. Flujo de desarrollo en Salesforce

De forma general, Salesforce da a sus clientes acceso a un entorno productivo a partir del cual se pueden crear, dependiendo del contrato firmado, diferentes tipos de sandbox [67].

Las modificaciones sobre el entrono productivo están limitadas para evitar causar problemas a los usuarios finales. Para desarrollar y realizar cambios se utilizan las sandbox sobre las que no aplican limitaciones de customización.

Una vez se ha desarrollado una solución en sandbox que se quiera desplegar sobre productivo, se crea un documento xml donde siguiendo un formato específico de Salesforce, se indican los elementos que conforman la solución. Dicha lista de elementos es empaquetada mediante la herramienta de comandos y se envía al entrono productivo para que el sistema valide su integridad.

La validación de integridad de la solución consta de dos partes. La primera es comprobar que los elementos contenidos del paquete no tienen dependencias con elementos inexistentes o desactualizados. Esto puede ocurrir cuando hay un cambio de versión entre los entornos o algún elemento ha quedado fuera del paquete. La segunda comprobación radica en la ejecución de las pruebas automatizadas de la plataforma para comprobar la integridad del código Apex. Este escenario solo es obligatorio para los despliegues en productivo, pero es aconsejable ejecutar las pruebas en entornos de sandbox para asegurarnos de cumplir con la cobertura mínima exigida por el sistema.

En caso de que falle alguna de las validaciones en productivo, el sistema rechaza el despliegue y no se produce ningún cambio.

Los despliegues pueden realizarse entre diferentes sandbox, lo que nos permite crear una especie de estructura espejo con respecto al repositorio de código, creando una sandbox por rama permanente y para las ramas de desarrollo en función del alcance de los cambios u otros requisitos de proyecto.

3.2.2. Modelado del flujo de desarrollo

Para la implementación de este trabajo no tenemos acceso a un entrono productivo por lo que nos es imposible crear sandbox. Existe la posibilidad de utilizar un DevHub [31] para generar sandbox temporales de forma programática, pero dada su naturaleza efímera, encajarlas dentro del pipeline tiene sus dificultades en cuanto autentificación y limitación de recursos.

Con la intención de simular una estructura de entrono productivo y sandbox utilizaremos tres Developer Edition [23] con roles específicos para cada una:

1. DE Producción
2. DE Preproducción
3. DE Desarrollo

Las organizaciones Developer Edition no están vinculadas entre ellas y no tienen limitaciones en cuanto a las modificaciones como es el caso de las organizaciones productivas. Sin embargo, mediante metadatos, podemos hacer despliegues entre ellas de la misma forma que en organizaciones vinculadas y modelar su comportamiento.

Como veremos en los apartados 3.3. y 3.10 de este capítulo con respecto al repositorio y al diseño de la implementación, el flujo de los cambios comenzará en DE Desarrollo, pasará por DE Preproducción y terminará en DE Producción.

En un pipeline más completa tendríamos entornos dedicados para QA, UAT y posiblemente Hotfix, pero es común tener un número de entornos limitados con los que trabajar, puesto que la cantidad y tipología de las sandbox disponibles va en función del contrato firmado con Salesforce. Tener menos entornos nos permitirá emular de forma mucho más ágil la ejecución del pipeline para la implementación que requiere este trabajo.

3.3. Estructura del repositorio

3.3.1. Herramienta elegida

Para la gestión del repositorio se utilizará la opción Cloud de GitHub [37] dado que es la herramienta elegida por la empresa en la que actualmente estoy trabajando.

Desde un punto de vista puramente técnico, no existen diferencias sustanciales respecto a las capacidades de automatización de Bitbucket [8] y GitHub. Ambas soluciones nos permiten implementar el pipeline que describo en este subapartado, salvando las diferencias en implementación intrínsecas a las dos herramientas.

La implementación se realizará utilizando la versión gratuita. Dado que algunas características son únicas de la versión Enterprise, que será la versión disponible para la compañía, se contemplará obtener una prueba de treinta días [38] para mostrar alguna de ellas, como sería la recuperación de los Audit Logs detallados del sistema.

Se creará una GitHub Action para alertar de modificaciones en tipologías de ficheros específicas como perfiles y conjuntos de permisos para forzar una validación por parte de los QA y el Líder técnico y funcional.

3.3.2. Estructura de ramas

En este subapartado planteo una estructura de ramas simplificada que se adecua a las organizaciones de Salesforce disponibles.

La solución de pipeline propuesta trabajará sobre un repositorio con tres tipologías de ramas:

Main → Rama única. Contendrá la versión estable desplegada en el entorno DE Producción de Salesforce. El líder técnico y funcional será el encargado de bloquear o aceptar los despliegues sobre esta rama.

Preproducción → Rama única. Contendrá la versión desplegada en el entorno de DE Preproducción de Salesforce. Los QA serán los encargados de validar las peticiones de despliegue de los desarrolladores sobre esta rama y su posterior validación.

Desarrollo → Habrá varias ramas de desarrollo. Todas las ramas de desarrollo deberán de estar alineadas con los entornos de desarrollo de Salesforce. Para esta implementación trabajaremos con una única organización de desarrollo, DE Desarrollo, y su correspondiente rama.

3.4. Protección de secretos

3.4.1. Protección de secretos en Salesforce

Salesforce tiene diversas formas de guardar información de configuración dependiendo del acceso que queramos permitir a dicha información. Para los datos de configuración públicos tenemos dos opciones:

1. **Custom Settings**

   Permite definir una serie de valores a nivel global y de forma específica para perfiles de usuarios concretos de modo que los valores para un mismo concepto varían en función de quién accede a la información [21]. El dato no puede encriptarse y es público.

2. **Custom Metadata Types**

   De forma similar a los Custom Settings, los Custom Metadata Types [20] nos permiten definir configuraciones. Sin embargo, estos no están limitados por perfiles y son más flexibles. El dato no puede encriptarse y es público.

Ninguna de estas dos opciones, por tanto, nos sirve para guardar secretos. Para proteger datos confidenciales tenemos otras dos opciones:

1. **Named and External Credentials**

   Los Named Credentials y External Credentials [51] permiten encapsular las integraciones de modo que los desarrolladores no requieren acceso a las credenciales de estas, gestionando todo el proceso de autenticación de forma separada a la ejecución del código.

   Los Named Credentials requieren de configuración a través de la interfaz de usuario. Tras una primera validación de la conexión, guardan las credenciales para futuras conexiones. No ofrecen capacidad de actualización automática de las credenciales más allá del refresco de los tokens en los casos en que estos se utilicen.

   La capacidad de refresco y establecimiento de secretos de los External Credentials es especifica a cada integración. Solo el sistema con el que se establece la integración puede realizar las acciones pertinentes para modificar la configuración de la credencial especifica, por lo que no pueden ser gestionados con una herramienta de terceros a través de la API.

   External Credentials es una capa añadida recientemente sobre los Named Credentials, de ahí que la documentación los trate como entidades separadas.

2. **Campos encriptados**

   Esta opción es, en realidad, un reaprovechamiento de las características de la plataforma que fueron implementadas con un objetivo diferente. Requiere de usar la base de datos principal como apoyo para el cifrado de datos sensibles. Hay dos niveles de cifrado, el que nos da la plataforma por defecto que utiliza un algoritmo de encriptación de 128-bits AES y el que nos ofrece la solución Shield de 256-bits AES [89]. Para gestionar el acceso a estos datos, se requiere utilizar el sistema de perfiles y roles de la base de datos, así como los conjuntos de permisos y perfiles.

   Esta opción requiere de una implementación customizada para su funcionamiento y tiene serias limitaciones en su opción gratuita, como el tamaño de la información a cifrar, la forma de acceder a ella y la imposibilidad de gestionar las claves de cifrado. Los secretos pueden ser desencriptados por los desarrolladores si pueden acceder a ellos a través del código, cosa que han de poder hacer para poder utilizarlos.

3.4.2. Propuesta de protección de secretos

La propuesta aquí planteada no se implementará en el contexto de este trabajo debido a que las organizaciones de tipo Developer Edition no vienen con la característica de Salesforce Shield activada.

Para una posible implementación necesitaríamos:

1. **Salesforce Shield**
   Dado que la solución es más completa y permite gestionar las claves de cifrado además de disponer de la API para la conexión entrante con AWS.

2. **AWS Secret Manager**
   Como herramienta de terceros con la que integrarnos mediante un External Credential para gestionar los secretos y la rotación de las claves de cifrado.

3. **Objeto Custom**
   El equivalente a una tabla en la base de datos donde guardar y cifrar los secretos mantenidos por AWS Secret Manager. Es necesario almacenar los secretos dentro de la plataforma Salesforce para aquellos escenarios en los que no puede realizarse una consulta API, como es el caso de los automatismos síncronos propios de Salesforce.

4. **Connected App**
   Las claves de cifrado se pueden rotar sin necesidad de disponer de una Connected App [15], pero la actualización de los secretos en la base de datos requerirá de una integración. Para ello se necesitará una Connected App que permita a AWS integrarse con la organización de Salesforce y modificar los datos en la tabla que contiene los secretos

El diseño de la solución se muestra en la Figura 4: **Diagrama de gestión de secretos**.

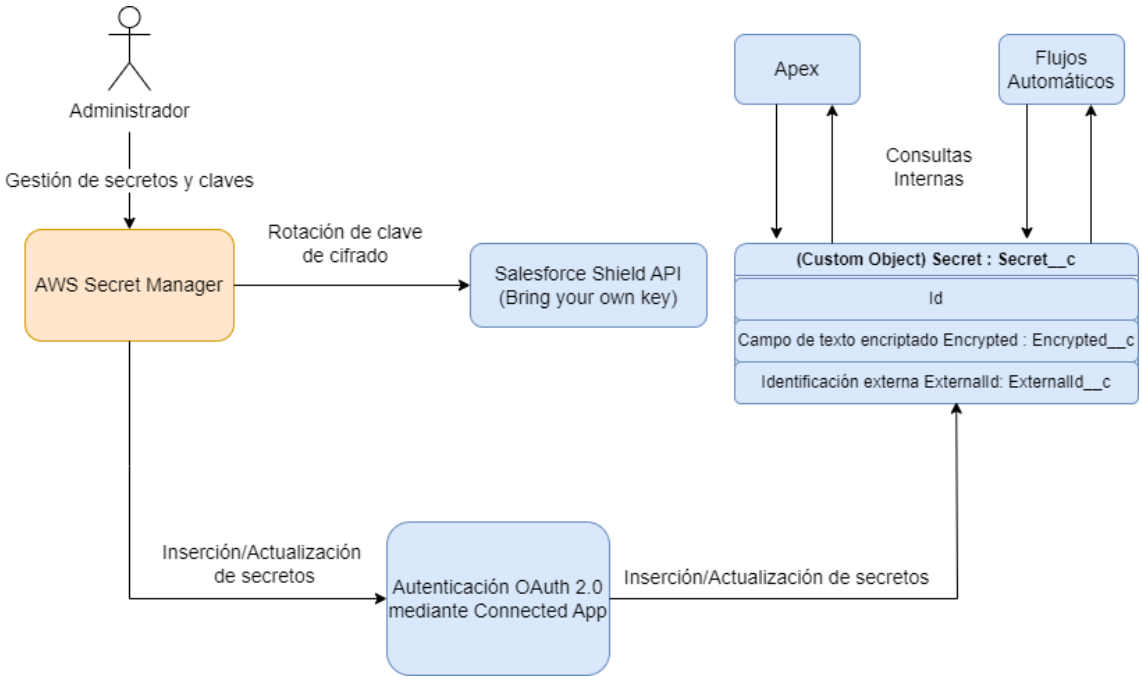


Figura 4: Diagrama de gestión de secretos
Fuente: Elaboración propia

3.5. Detección de secretos

3.5.1. Herramienta elegida

Como ya he comentado en el subapartado 3.4.1, los secretos de Salesforce se gestionan de forma separada al repositorio de código. No obstante, siempre queda la posibilidad de que se puedan subir accidentalmente a las ramas secretos a nivel de código, o contenidos en otro tipo de ficheros durante el desarrollo. Con tal de evitar estos escenarios, las subidas a las ramas de desarrollo serán analizadas mediante GitLeaks [39] dado que no se requiere de la complejidad ofrecida por GitGuardian [36].

Utilizo GitHub Actions para lanzar el análisis.

El principal motivo de mantener la detección de secretos separada de la herramienta de análisis estático del código reside en el coste y flexibilidad del análisis. Como veremos en el siguiente apartado, utilizaremos SonarCloud [70], cuyo coste económico va en función del número de líneas de código a analizar. Si quisiéramos utilizar su capacidad de detección de secretos, deberíamos incluir todos los ficheros del repositorio en el análisis, incrementando sustancialmente el número de líneas y, por tanto, el coste [73].

3.6. Análisis estático del código

3.6.1. Herramienta elegida

Para el análisis estático del código he elegido SonarCloud/SonarLint por los siguientes motivos:

1. Puede analizar Apex de forma nativa sin necesidad de tener que construir el conjunto de reglas desde cero.
2. Ofrece análisis para JavaScript, Html y Css también de forma nativa, con lo que podemos dar cobertura a la parte del frontend.
3. Clean as you code nos ayuda a trabajar con una base de código ya existente que no cumple con las normativas a implementar para irla actualizando poco a poco.
4. Al ir de la mano con SonarLint, ello nos permite utilizar las mismas reglas de análisis de código en máquinas locales sin tener que mantener dos conjuntos de reglas separadas.

3.6.2. Configuración

Restringiremos el análisis de código a los ficheros de JavaScript, CSS, Html y clases de Apex. Dejaremos fuera del análisis los ficheros de metadatos en los cuales se

define la configuración de los objetos, campos y otros elementos del sistema no relacionados con el código.

Las razones de ello son:

1. El coste de SonarCloud va en función del número de líneas de código a analizar y los metadatos tienen muchas líneas que no son relevantes.
2. Los ficheros de metadatos representan la configuración de la organización y, aunque pueden ser fuente de manipulaciones maliciosas, su detección implicaría un análisis caso a caso para identificar que modificaciones son válidas y cuáles no. Algo que nos permite hacer de forma mucho más ágil el GitHub con GitHub Actions.

Los resultados del análisis se mostrarán en GitHub y será responsabilidad de los QAs y el Líder técnico y funcional marcar los falsos positivos, así como rechazar las peticiones que no cumplan con las reglas definidas.

Añadiré algunas reglas adicionales para demostrar cómo reacciona e informa de ellas el sistema.

3.7. Escaneo de contenedores

Para la gestión automática de los despliegues, utilizaremos un contenedor Docker en el que cargaremos la herramienta de línea de comandos de Salesforce. Esta imagen de Docker se ejecutará con cada modificación sobre las ramas de Preproducción y Producción para realizar un despliegue sobre el entrono respectivo utilizando una Connected App para regular el acceso y la autenticación. Para ello tomará la última versión de la rama y ejecutará los comandos para su despliegue. En caso de error informará de ello a través del repositorio. Usaremos esta misma imagen para la ejecución de las pruebas DAST.

Aunque la imagen del contendor no es manipulada por los desarrolladores, nos hemos de asegurar que los secretos necesarios para integrarnos con Salesforce a través de la Connected App están correctamente protegidos y que no hay vulnerabilidades que se puedan aprovechar para causar modificaciones maliciosas o no intencionadas sobre los entornos de Salesforce.

3.7.1. Herramienta elegida

Docker Scout [26] es la herramienta elegida por su flexibilidad y la posibilidad de mostrar resultados a través de GitHub. No formará parte del ciclo de desarrollo, pero se aplicará a cada nueva versión de la imagen de gestión de despliegues antes de subirla y activarla para el repositorio.

3.8. Análisis dinámico de aplicaciones

3.8.1. Limitaciones de Salesforce

Los análisis dinámicos de las aplicaciones sobre el software base están restringidos por el fabricante, que ofrece sus propias garantías de seguridad. Por ello, este tipo de análisis debe limitarse a los elementos añadidos sobre el software de base.

Las pruebas que podemos realizar están, a su vez, limitadas.

Para pruebas de análisis intensivo hay que solicitar permiso con antelación con tal de evitar sobrecargar los servidores. Salesforce comparte sus servidores con varios clientes de forma simultánea, asignando los recursos disponibles entre los participantes. Si un participante trata de bloquear una cantidad de recursos desproporcionada, como ocurriría durante una prueba de análisis intensivo, Salesforce bloquea al participante para preservar el correcto funcionamiento de los servidores. Notificarles con anterioridad les permite introducir una excepción a esta regla durante el periodo de las pruebas. Ello nos impide automatizar este tipo de pruebas.

Además, Salesforce ofrece seguridad a nivel de interfaz de usuario de base mediante Lightning Web Security [50] que sigue los estándares TC39 [83]. En esencia, este modelo de seguridad limita el uso de librerías JavaScript no seguras, bloquea ataques de cross-site scripting y restringe ciertas funcionalidades de acceso al DOM que pueden causar vulnerabilidades.

Como veremos en el apartado 4.8 estas limitaciones me han impedido automatizar los procesos de análisis para los servicios.

3.8.2. Herramienta elegida

He elegido Zed Attack Proxy [91] para la implementación del análisis DAST por su compatibilidad con GitHub Actions y capacidad de adaptación. A pesar de que Invicti tiene muchas más funcionalidades, la naturaleza de Salesforce hace que muchas de ellas sean inaplicables.

Automatizaremos el proceso de análisis de esta herramienta para los entornos de preproducción y producción como parte de la imagen Docker utilizada para los despliegues. El análisis sobre preproducción y producción se realizará después de los despliegues en los entornos, dado que no puede realizarse el análisis con anterioridad. Los resultados se mostrarán en el repositorio.

Para soluciones futuras, se podría contemplar un proceso de automático de Roll Back para casos en que la herramienta DAST detecte errores.

3.9. Cumplimiento normativo

Como ya he mencionado en el apartado 2.7, para el cumplimiento normativo quiero centrarme en mostrar la actividad de los usuarios finales y administradores en busca de comportamientos anómalos.

Como veremos en el apartado 4.9 no he sido capaz de configurar la herramienta para extraer mediante integración y mostrar los datos de los Audit Logs.

3.9.1. Herramienta elegida

Para la implementación de este trabajo utilizo OpenSearch [53] debido a las limitaciones de Splunk con sus versiones de prueba. Dado que la mayoría de las herramientas seleccionadas carecen o no requieren de Audit Logs, me centraré en recuperar la información sobre los sistemas de GitHub y Salesforce a través de sus respectivas API. En el caso de Salesforce, necesitaremos de una Connected App específica para establecer la conexión.

3.10. Diagramas del pipeline

3.10.1. Diagrama Desarrollador

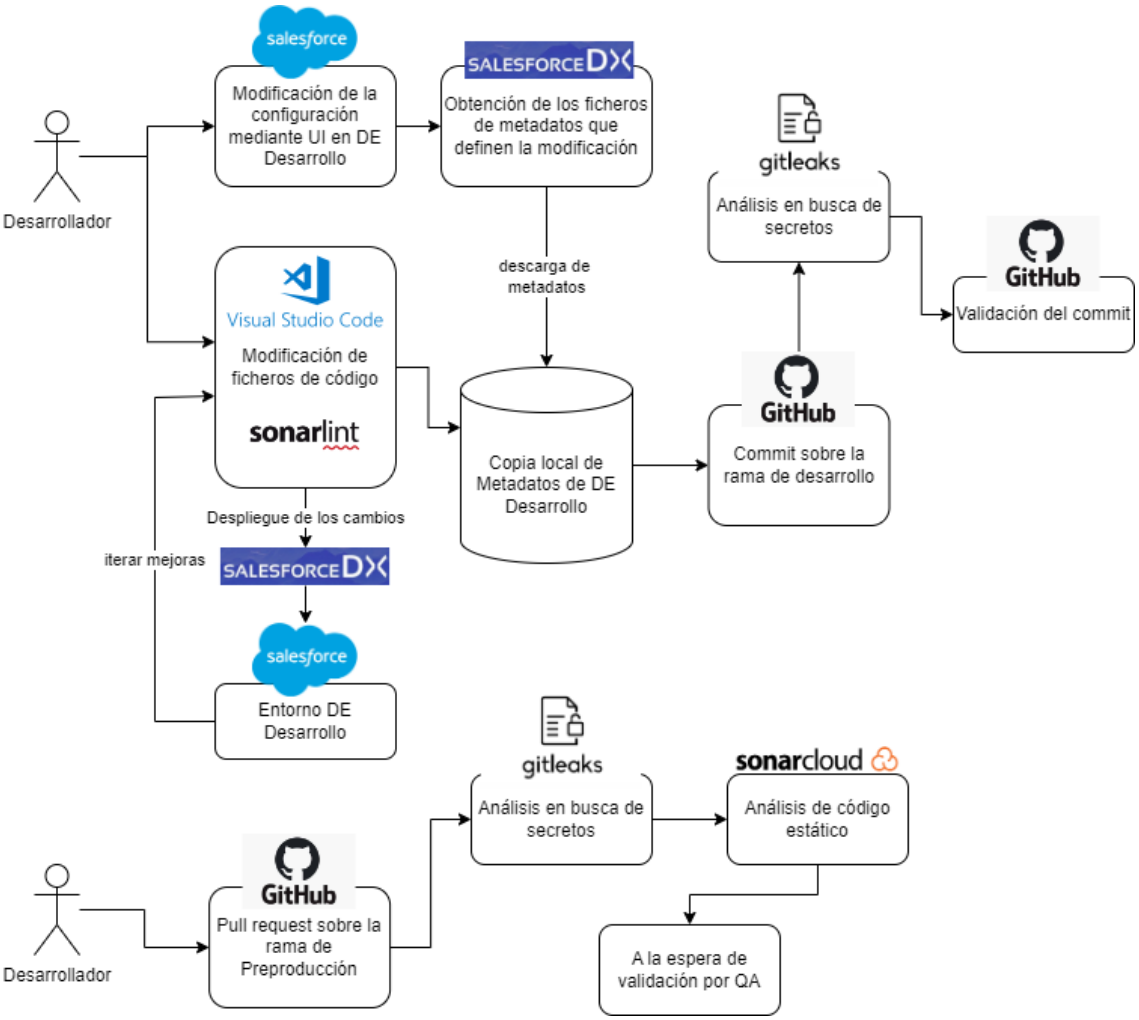


Figura 5: Diagrama de flujo del desarrollador

Fuente: Elaboración propia

Los desarrolladores realizarán las modificaciones necesarias sobre el entorno de DE Desarrollo utilizando la interfaz de Salesforce para las modificaciones de configuración y Visual Studio Code con el plugin de SonarLint para las modificaciones de código. Mediante Salesforce DX, los desarrolladores harán despliegues de código sobre el entorno de DE Desarrollo y descargarán las configuraciones manuales de modo que, al hacer commit sobre la rama de desarrollo en la que estén trabajando, se suban las definiciones de los dos tipos de modificaciones al repositorio.

Para cada commit se ejecutará la GitHub Action de GitLeaks.

Una vez los desarrolladores consideren que un conjunto de cambios está listo para su promoción al entrono de DE Preproducción para que lo validen los QAs, procederán a crear una pull request sobre la rama de Preproducción.

Sobre la pull request creada ejecutaremos la GitHub Action de GitLeaks y el análisis de SonarCloud. La aprobación de los cambios y la continuación del flujo de despliegue quedará en manos de QA.

3.10.2. Diagrama QA

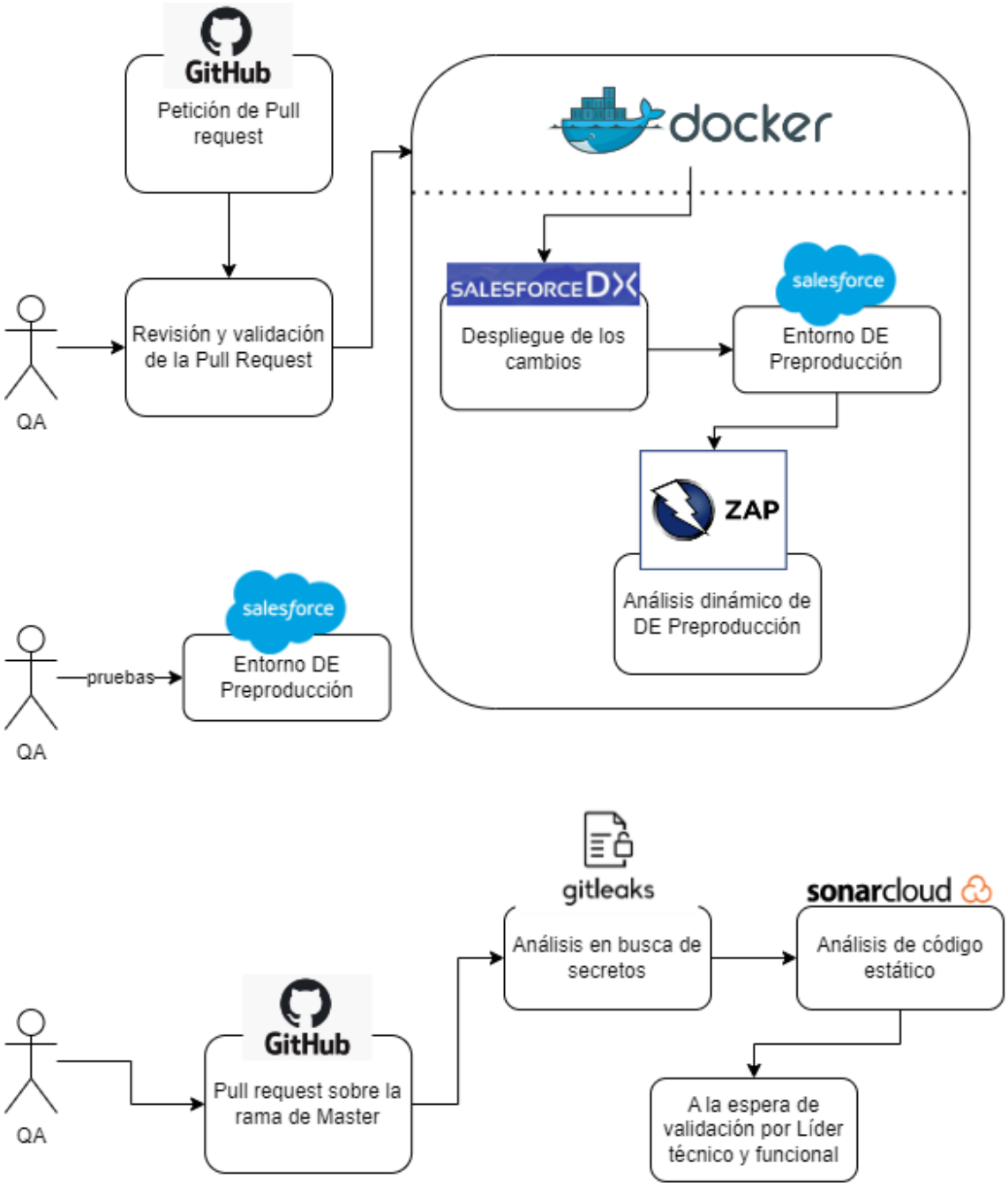


Figura 6: Diagrama de flujo del QA

Fuente: Elaboración propia

El QA revisará las peticiones de pull request sobre la rama de Preproducción. En caso de considerar que los cambios son válidos, dará el visto bueno y se procederá al merge con la rama de Preproducción. Tras el merge, se ejecutará de forma automática

la GitHub Action que lanzará la imagen de Docker configurada para ejecutar el despliegue de los cambios sobre DE Preproducción seguido por las pruebas DAST.

En el caso que los resultados sean correctos, se procederá a realizar un pull request sobre la rama Main (este paso no es automático puesto que algunas de las pruebas de QA y UAT hay que realizarlas sobre el entorno de DE Preproducción). Al igual que con el pull request sobre la rama de Preproducción, ejecutaremos la GitHub Action de GitLeaks y el análisis de SonarCloud. La aprobación de los cambios y la continuación del flujo de despliegue quedará en manos del Líder técnico y funcional.

3.10.3. Diagrama Líder técnico y funcional

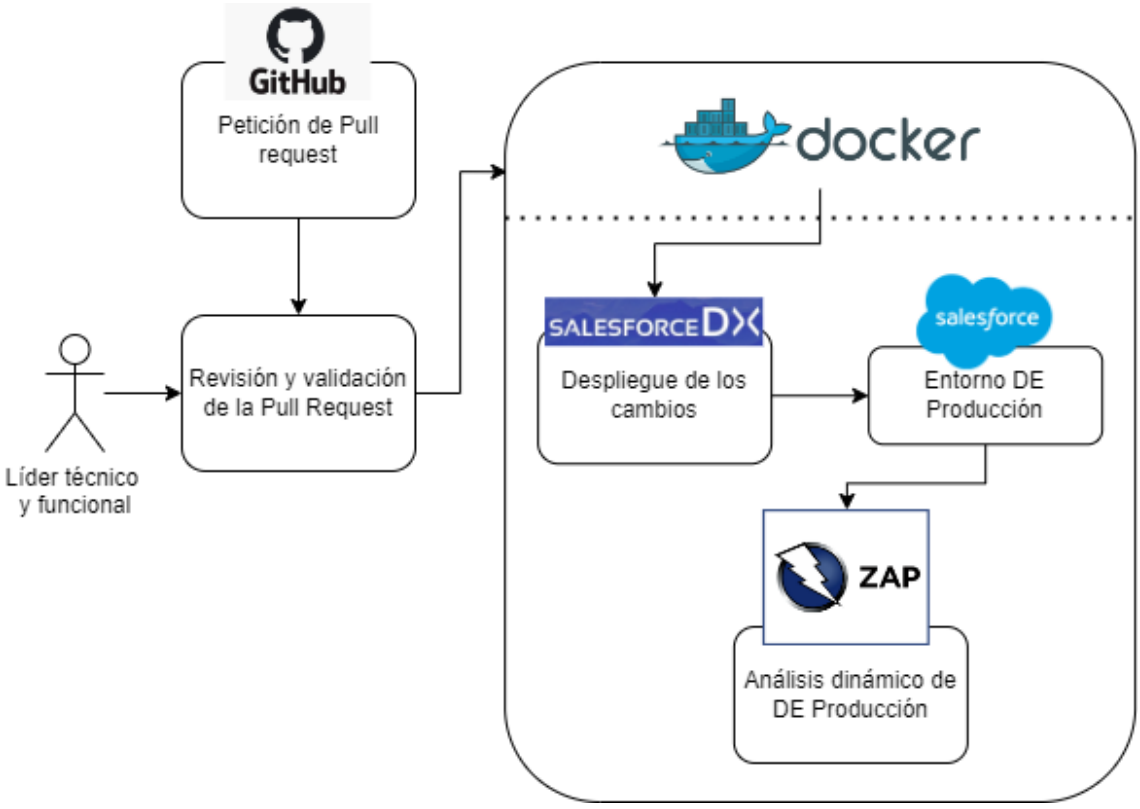


Figura 7: Diagrama de flujo del Líder técnico y funcional
Fuente: Elaboración propia

El Líder técnico y funcional revisará las peticiones de pull request sobre la rama de Main. En caso de considerar que los cambios son válidos, dará el visto bueno y se procederá al merge con la rama Main. Al igual que con Preproducción, tras el merge, se ejecutará de forma automática la GitHub Action con la imagen de Docker para el despliegue sobre DE Producción y el posterior análisis DAST.

3.10.4. Diagrama Administrador

Los administradores se encargarán del mantenimiento y la modificación de la configuración de las distintas herramientas del pipeline, así como del alta y baja de los usuarios finales.

Hay dos herramientas que los Administradores utilizarán:

1. Docker Scout, para analizar las nuevas versiones de las imágenes Docker que utilizan las GitHub Actions.
2. OpenSearch, para recuperar los logs de actividad de Salesforce y GitHub.

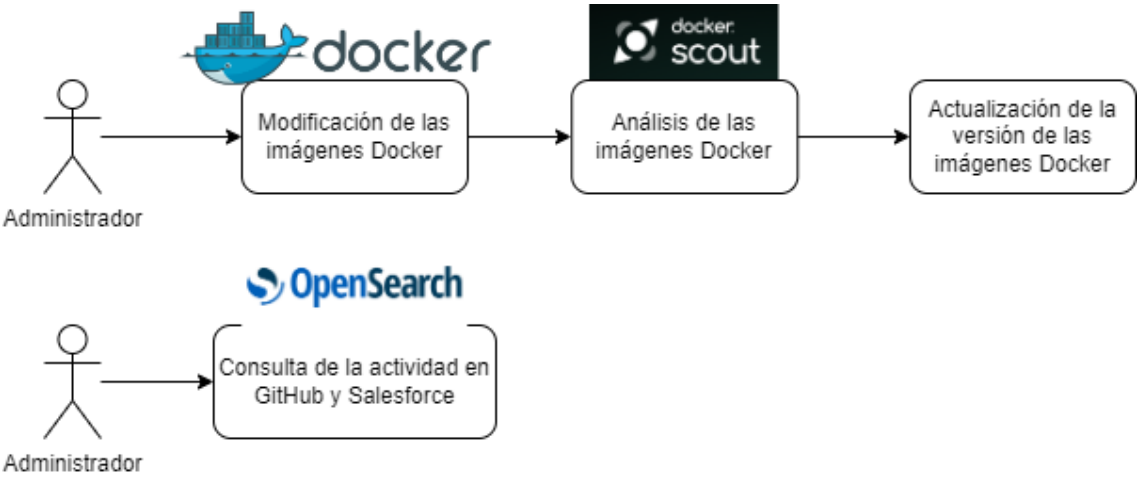


Figura 8: Diagrama de flujo del Administrador

Fuente: Elaboración propia

3.10.5. Diagrama Administrador de Sistemas

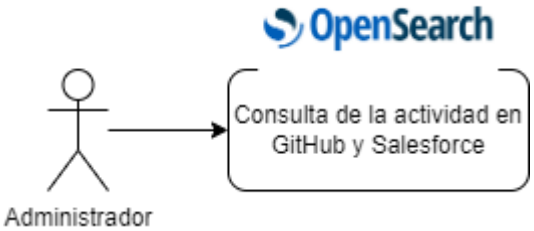


Figura 9: Diagrama de flujo del Administrador de sistemas

Fuente: Elaboración propia

El Administrador de sistemas tiene dos responsabilidades:

1. Dar de alta/baja administradores.
2. Mantener las cuentas en marcha manteniendo/adquiriendo las licencias necesarias.

El Administrador de Sistemas utilizará OpenSearch para monitorizar las acciones de los Administradores.

4. Implementación del pipeline

4.1. Usuarios

Para mantener los diferentes usuarios separados y que no haya conflictos durante las configuraciones, creo cinco cuentas de correo en Gmail [40]:

1. Desarrollador → desarrollador.tfm.jmgs@gmail.com
2. QA → qa.tfm.jmgs@gmail.com
3. Líder técnico y funcional → lider.tfm.jmgs@gmail.com
4. Administrador → atmin.tfm.jmgs@gmail.com
5. Administrador de sistemas → sysatmin.tfm.jmgs@gmail.com

Con el objetivo de facilitar el trabajo y evitar solapamiento de las sesiones de usuario trabajando con las herramientas en el cloud, asigno un navegador diferente a cada usuario:

1. Desarrollador → Opera GX [57]
2. QA → Opera [56]
3. Líder técnico y funcional → Chrome [10]
4. Administrador → Firefox [33]
5. Administrador de sistemas → Edge [29]

4.2. Salesforce DE

Siguiendo lo indicado en el capítulo 3, creo tres Developer Edition con los siguientes usuarios:

1. DE Producción
    a. Administrador de sistemas
        i. sysatmin.tfm.jmgs@gmail.com
    b. Líder técnico y funcional
        i. lider.tfm.jmgs@gmail.com
2. DE Preproducción
    a. Administrador de sistemas
        i. sysatmin.tfm.jmgs@gmail.com.pre
    b. QA
        i. qa.tfm.jmgs@gmail.com.pre
3. DE Desarrollo
    a. Administrador de sistemas
        i. sysatmin.tfm.jmgs@gmail.com.dev
    b. Desarrollador
        i. desarrollador.tfm.jmgs@gmail.com.dev

El administrador de sistemas está presente en todas las organizaciones dado que es él quién las da de alta. No creo administradores para las organizaciones porque las Developer Edition restringen el número de usuarios. En un escenario como el que planteo y menciono en el capítulo 3, sería necesario crear a los administradores para separar las responsabilidades, pero dada la limitación me veo obligado a tener solo administrador de sistemas y crear con ellos los perfiles y permisos necesarios.

Ver anexo 1 para los detalles de los procesos de alta y configuración de las cuentas, perfiles y permisos.

Perfil: Tech Support clonado a partir del perfil Standard User.
Permission Set: Permisos de desarrollo, con los siguientes permisos habilitados:

- API Enabled
- Modify All Data
- Author Apex
- View Setup and Configuration
- Customize Application

4.3. Configuración básica de Visual Studio Code

Como he mencionado con anterioridad, para que el desarrollador realice su labor, necesito configurar Visual Studio Code en su ordenador de trabajo. En este apartado cubro la configuración básica. En los siguientes apartados cubro las configuraciones específicas de las herramientas de seguridad que interactúan con Visual Studio Code.

Para tener la base de trabajo con Visual Studio Code es necesario:

- Visual Studio Code [87]
- Java 17/22 [46]
- Node [52]
- Salesforce CLI [63]
- Paquete de extensiones de Salesforce [66]

Para la instalación he utilizado el ejecutable de Visual Studio Code de la versión 1.89.1. x64 para el sistema operativo Windows 11. He seguido los pasos indicados por el programa de instalación dejando los valores por defecto. Como tema de la aplicación he seleccionado Dark modern.

Tras la instalación de Visual Studio Code, procedo a la instalación de Node y Java, así como de la CLI de Salesforce x64. Selecciono todos los componentes disponibles para su instalación y dejo el resto de los valores por defecto. En las siguientes figuras vemos las versiones instaladas.

```
C:\Users\josem>sf -v
@salesforce/cli/2.42.6 win32-x64 node-v20.12.2
```

Figura 10: Versión de SF CLI y node
Fuente: elaboración propia

Instalo el paquete de extensiones versión 60.15.0 que contiene:

- Apex
- Apex interactive debugger
- Apex replay debugger
- Salesforce CLI integration
- Aura components
- Visualforce
- Lightning web components
- SOQL
- SLDS Validator

Creo un proyecto de Salesforce utilizando el atajo de comandos que las extensiones han añadido en Visual Studio Code.

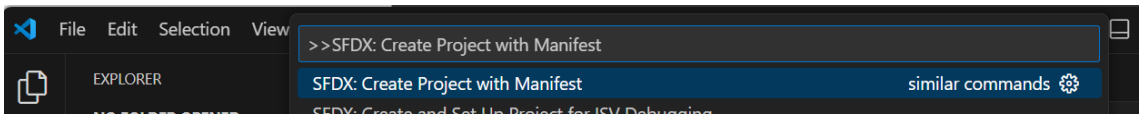


Figura 11: Crear proyecto de Salesforce con manifiesto
Fuente: elaboración propia

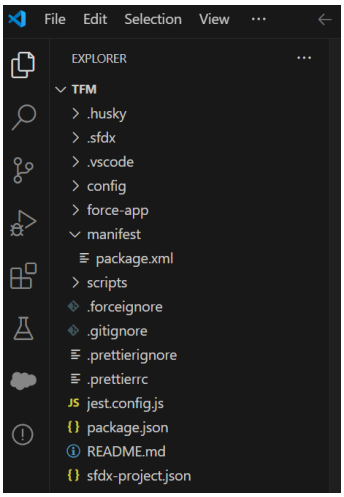


Figura 12: Proyecto TFM
Fuente: elaboración propia

Conecto Visual Studio Code con la organización de desarrollo utilizando el atajo de comandos de las extensiones.

Figura 13: Autorizar una organización
Fuente: elaboración propia

Cuando me pide el tipo de url a introducir, selecciono custom y utilizo la url de la organización de desarrollo:

https://uoctfmciberseguridad3-dev-ed.develop.my.salesforce.com

Se abre el navegador por defecto del sistema para proceder a la autentificación. Uso las credenciales del desarrollador.

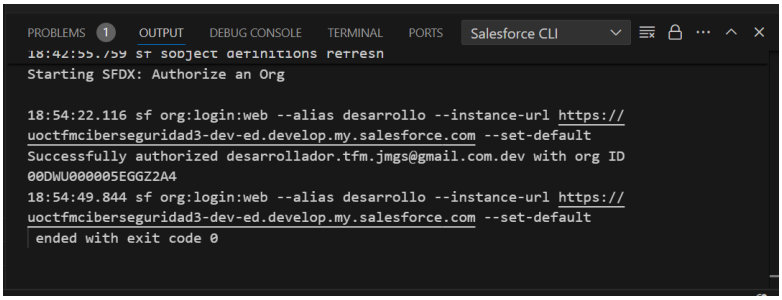


Figura 14: Resultado de la autorización
Fuente: elaboración propia

Con esto, el desarrollador puede programar en local y subir los cambios que realice a la organización mediante las acciones de despliegue facilitadas por las extensiones o utilizando el terminal para ejecutar los comandos de sfdx.

4.4. Configuración de GitHub

Para la creación de las cuentas de GitHub, ver el anexo 2. Tengo cinco cuentas:

1. Desarrollador → desarrollador-tfm-jmgs
2. QA → qa-tfm-jmgs
3. Líder técnico y funcional → líder-tfm-jmgs
4. Administrador → atmin-tfm-jmgs
5. Administrador de sistemas → sysatmin-tfm-jmgs

Las cuentas tienen los permisos necesarios para realizar las acciones indicadas en el apartado 3.10, con la salvedad que las aprobaciones de las pull request de preproducción pueden ser validadas por el líder técnico y funcional además del QA.

Utilizo una cuenta Enterprise de prueba de treinta días asociada a la cuenta del administrador de sistemas debido a que solo este tipo de cuentas pueden utilizar Environment Secrets. Esta característica es necesaria para restringir el acceso a los secretos del repositorio con el que trabajaremos.

4.4.1. Configuración base del repositorio

Procedo a la creación del repositorio llamado tfm dentro de la organización TFM-UOC-JMGS-Ciberseguridad con la siguiente configuración:

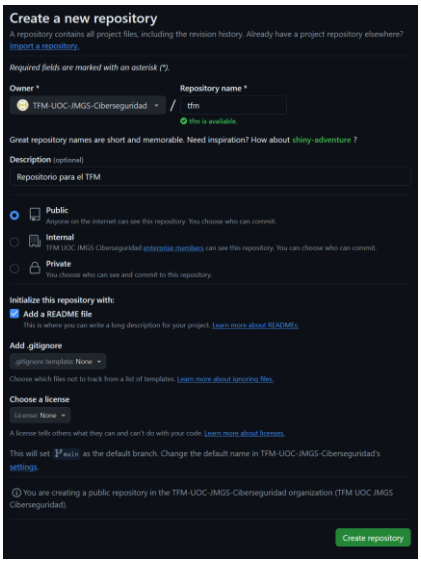


Figura 15: Creación del repositorio
Fuente: elaboración propia

Configuro tres roles para el repositorio:

- Desarrollador → Hereda del rol estándar Write.
- QA → Hereda del rol estándar Write.
- Líder técnico y funcional → Hereda del rol estándar Maintain.

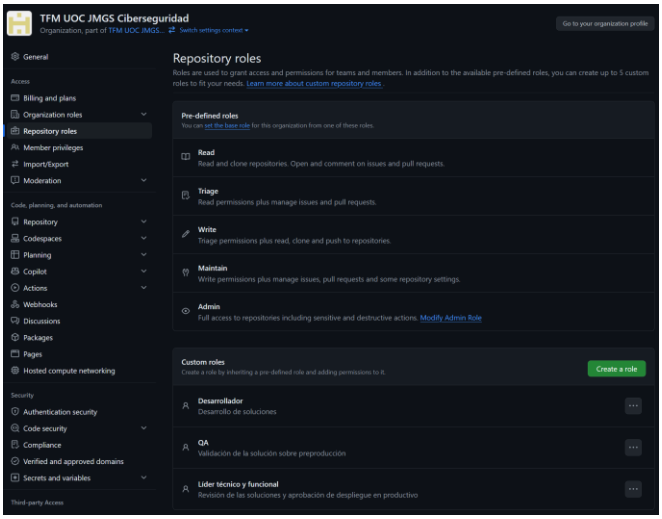


Figura 16: Roles del repositorio
Fuente: elaboración propia

Configuro cuatro equipos para la organización:

- Equipo de administración → equipo-admin
- Equipo de producción → equipo-pro

- Equipo de preproducción → equipo-pre
- Equipo de desarrollo → equipo-dev

A cada equipo añado a los usuarios correspondiente. El administrador formará parte de todos los equipos por si se requiere de su intervención para solucionar alguna incidencia. En una organización real, habrás más de un usuario en cada equipo y los usuarios entrarán y saldrán de los equipos. Será sobre los equipos donde configuraré las reglas de protección y propiedad del código para agilizar los procesos de alta y baja.

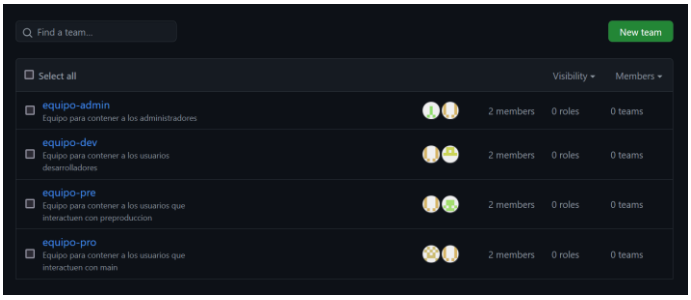


Figura 17: Equipos GitHub
Fuente: elaboración propia

Creo dos ramas, preproduccion y desarrollo-inicializacion. Preproduccion será una rama protegida, mientras que desarrollo-inicializacion creada a partir de preproducción será utilizada para cargar la base del proyecto de Salesforce en el repositorio sobre la que comenzar a trabajar. A lo largo de las pruebas iré creando ramas de desarrollo en función de las necesidades.

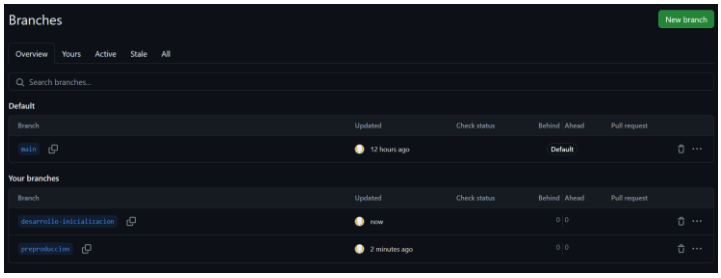


Figura 18: Ramas del repositorio
Fuente: elaboración propia

4.4.2. Protección de las ramas

Creo las reglas de .github\CODEOWNERS [12] para limitar quién puede aprobar los cambios sobre los ficheros del repositorio. La definición del fichero de codeowners sigue un patrón de la última regla tiene prioridad, por lo que comienzo dando la propiedad de todos los ficheros del repositorio al equipo de administración y luego fijo la propiedad de los ficheros de la solución al equipo de producción. De este modo el fichero de codeowners y otros ficheros de configuración, como los de las GitHub Actions quedan bajo la autoridad del equipo de administración.

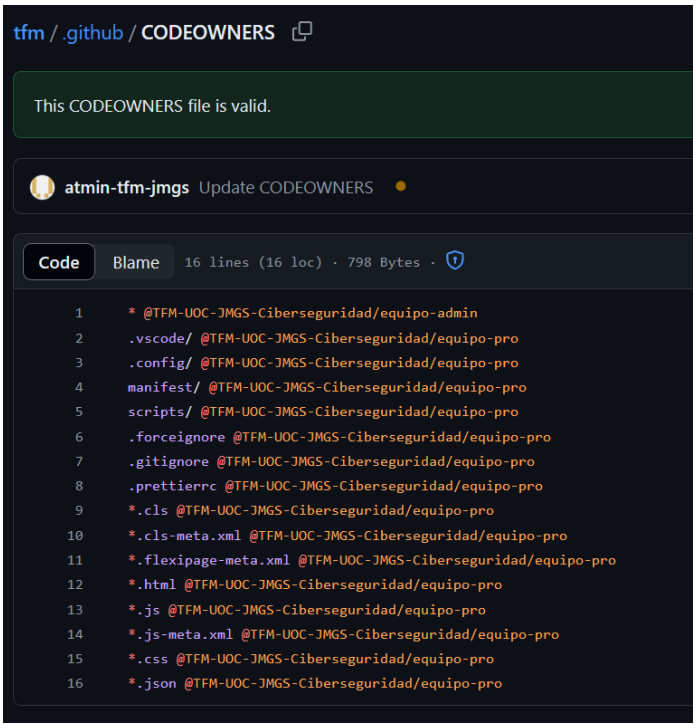


Figura 19: Fichero de CODEOWNERS de GitHub

Fuente: elaboración propia

En este punto me encuentro con un problema sobre el que la documentación de GitHub no aporta información. El documento de codeowners aplica, tal como indica la documentación oficial, a cada rama del repositorio por separado. Sin embargo, el fichero en si forma parte del repositorio junto al resto de ficheros, por lo que no es posible mantener copias del fichero diferenciadas por rama sin tener conflictos con las operaciones de merge, clonado, etc.

Las soluciones que he encontrado a este problema son las siguientes:

- Formar a los equipos para que eviten incluir el fichero de codeowners en las pull request.
- Crear automatismos para falsificar las modificaciones del fichero.

Tanto una como otra opción conllevan altos riesgos de seguridad, por lo que he decidido reservar la configuración de codeowners solo para restringir las validaciones en productivo. El resto de las ramas contendrán el mismo fichero de codeowners, pero no será obligatorio requerir de la validación de los propietarios de los ficheros para subir cambios hasta el entorno de preproducción.

Procedo a añadir a los equipos al repositorio con sus respectivos roles:

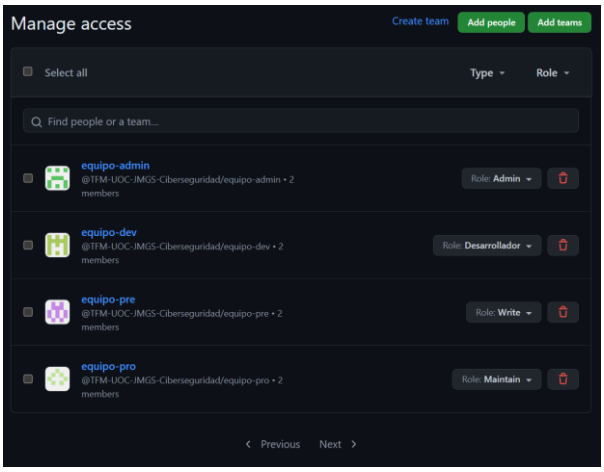


Figura 20: Usuarios del repositorio
Fuente: elaboración propia

Configuro las reglas de protección para las ramas de main y preproducción. Añado el equipo de preproducción a la regla de protección de preproducción y al de producción a la de main de modo que sean los únicos equipos que puedan rechazar las pull request en cada rama especifica. Marco la opción 'Require review from code owners' solo en la regla de producción obligar a los equipos de administración y producción a aprobar los cambios de los ficheros para los que tienen ownership.

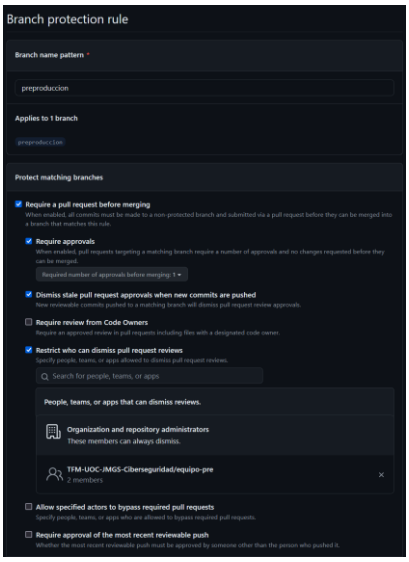


Figura 21: Regla de protección de pre
Fuente: elaboración propia

Finalmente, realizo los commits iniciales contra desarrollo con el código de un proyecto de Salesforce vacío. Para ver los pasos realizados de la conexión con el Visual Studio Code y el commit, consultar el anexo 3.

Ahora, a partir del commit inicial sobre desarrollo, creamos la pull request desde la interfaz de GitHub con el usuario del desarrollador para subir estos cambios a la rama de preproduccion.

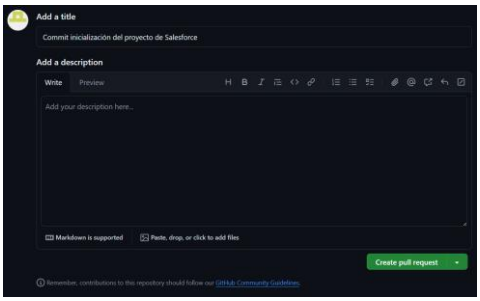


Figura 22: Creación de la pull request
Fuente: elaboración propia

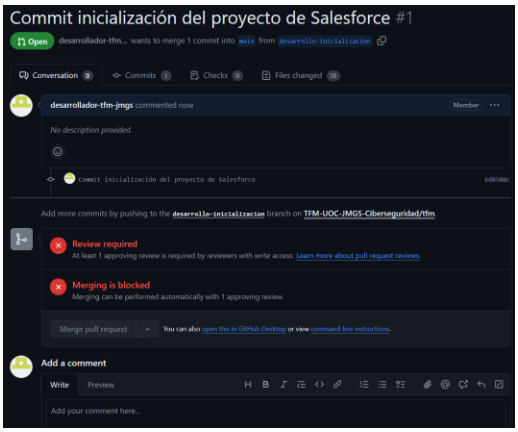


Figura 23: Pull request pendiente de aprobación
Fuente: elaboración propia

Entro con el usuario de QA para validar la pull request y hago el merge de la petición con la rama de preproducción.

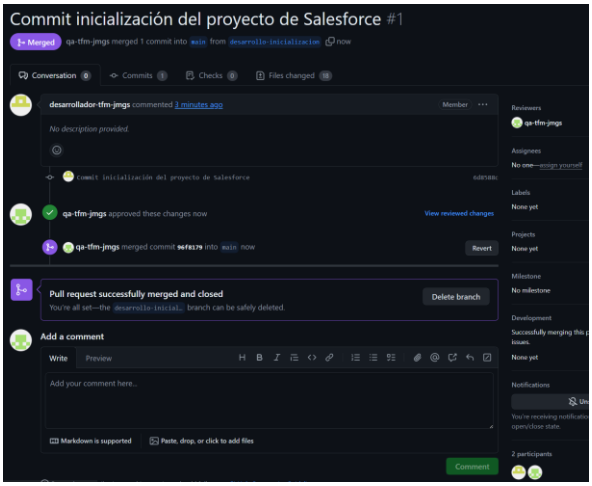


Figura 24: Pull request validada por el QA y merge
Fuente: elaboración propia

Procedo a utilizar el usuario de QA para solicitar una pull request contra la rama main y al usuario del líder técnico y funcional para validarla y hacer el merge.

Con esto tenemos el repositorio preparado para configurar el resto de las herramientas.

4.4.3. Automatización: GitHub Environments

Para automatizar los despliegues utilizaremos una GitHub Action que ejecute una imagen de Docker en un contenedor. La imagen de Docker contendrá las instrucciones necesarias para ejecutar el despliegue.

Los secretos necesarios para ejecutar las instrucciones de despliegue serán gestionados por GitHub. Utilizaremos dos environments, uno para preproducción y otro para producción con tal de contener los secretos y protegerlos de accesos no autorizados.

Para el environment de preproducción, añado el equipo-pre como equipo para validar los accesos a los secretos y marco la opción de requerir la revisión de acceso. Indico que este environment solo aplica para los accesos realizados desde la rama de preproduccion. Los secretos que crear son:

- SALESFORCE_URL
  - La url del entorno de preproducción o producción.
- SALESFORCE_JWT
  - La clave del jwt para conectarnos con la Connected App de Salesforce correspondiente a preproducción o producción.
- SALESFORCE_ALIAS
  - El alias de preproducción o producción.
- SALESFORCE_CLIENT_ID
  - El valor de consumer key de la Connected App
- SALESFORCE_USER
  - El usuario con el que conectaremos a través de la Connected App.

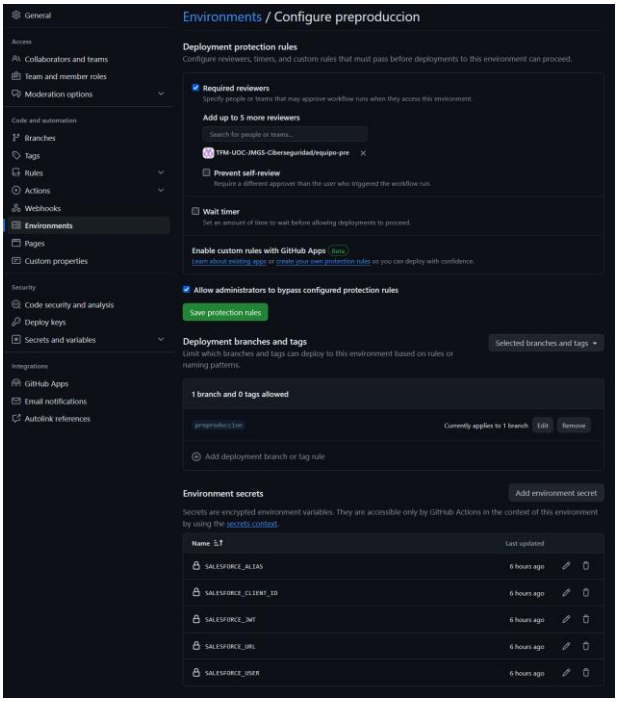


Figura 25: GitHub Environment preproducción
Fuente: elaboración propia

4.4.4. Automatización: Salesforce Connected App

He seguido los pasos indicados en la documentación de Salesforce [17] para la creación de las Connected App. Como punto a tener en cuenta necesitamos a un usuario porque la instrucción de “org login jwt” no soporta el uso de Client Credentials Flow por el momento.

Para la creación del certificado he utilizado Openssl [55] en un ordenador con macOS Catalina ejecutando las instrucciones de [18]:

- openssl genpkey -des3 -algorithm RSA -pass pass:**** -out server.pass.key -pkeyopt rsa_keygen_bits:2048
- openssl rsa -passin pass:**** -in server.pass.key -out server.key
- openssl req -new -key server.key -out server.csr
- openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt

El fichero que cargar en la Connected App es server.crt. El contenido del fichero que cargar en el secreto del repositorio es server.key.

4.4.5. Automatización: Docker GitHub Actions

Esta es la estructura que he elegido para gestionar las GitHub Actions y los Dockerfiles:

- Los ficheros .yml que definen la GitHub Actions están almacenados en la carpeta por defecto de .github/workflows/ junto a la Action de Gitleaks. Tendré una acción para producción y otra para preproducción. Aplicaré una u otra en función de la rama sobre la que se hace el despliegue.
- Las carpetas que contienen el fichero Dockerfile y el .yml de entrada para cada acción estarán contenidas en la carpeta docker-actions.

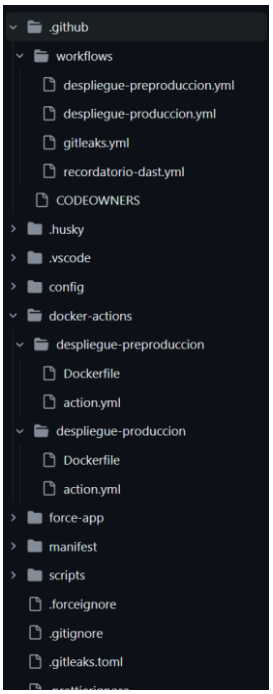


Figura 26: Estructura de ficheros GitHub Docker Actions

Fuente: elaboración propia

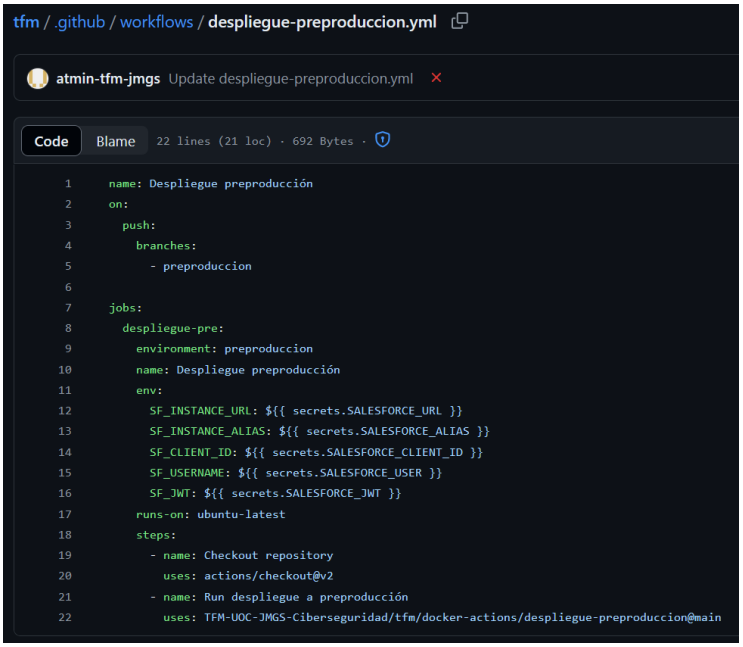


Figura 27: GitHub Action para ejecutar el contenedor Docker

En la figura anterior se puede ver el contenido de la GitHub Action de preproducción. En la sección de on indico que solo quiero lanzar la acción cuando se produzca un push sobre la rama de preproducción. En la sección de jobs, dentro de env, accedo a los secretos para cargarlos en las variables de entorno del contenedor.

No estoy utilizando el almacenamiento de los secretos de Docker porque las imágenes no son referenciadas en ningún otro contexto más que en el que se ejecutan.

En la sección de steps, realizo un checkout del repositorio para que el contenedor tenga acceso al proyecto de Salesforce mediante la GitHub Action de checkout estándar y luego procedo a llamar al .yml que indica el Dockfile a ejecutar, en concreto, la versión del .yml de la rama de producción (no la de preproducción).

Los Dockerfile a ejecutar contendrán una única instrucción para lanzar la imagen correspondiente almacenada en DockerHub. De ese modo, las instrucciones de despliegue a ejecutar no quedan expuestas en el repositorio.

Figura 28: Dockerfile preproducción
Fuente: elaboración propia

Figura 29: Dockerfile producción
Fuente: elaboración propia

La imagen a la que hacen referencia ha sido generada a partir del siguiente Dockerfile y el archivo sf-dx-commands.sh.

Figura 30: Dockerfile imagen
Fuente: elaboración propia

Utilizo la imagen de Docker para la CLI de Salesforce como base, copio el fichero donde tengo las instrucciones a ejecutar y lo utilizo como punto de entrada. El propósito de cada línea se explica en el printf previo a la línea en la siguiente figura:

Figura 31: Archivo sf-dx-commands.sh
Fuente: elaboración propia

4.5. Configuración de GitLeaks

Para utilizar la GitHub Action de GitLeaks, es necesario solicitar una licencia para la organización [49]. La licencia la solicita el administrador de sistemas y la utiliza el administrador que es quién tiene acceso a la organización que ha creado en GitHub.

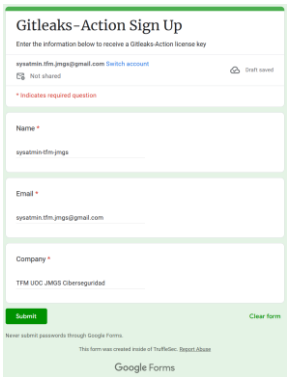


Figura 32: Formulario de licencia GitLeaks-Action
Fuente: elaboración propia

Para la configuración de la GitHub Action de GitLeaks sigo el ejemplo de [49]. Quiero que la Action se lance en los push sobre las ramas de desarrollo y en las pull request. Como necesito la licencia de GitLeaks referenciada en el fichero yml que define la GitHub Action, utilizo la funcionalidad de gestión de secretos de GitHub y creo un secreto a nivel de organización llamado GITLEAKS_LICENSE disponible para todos los repositorios de la organización.

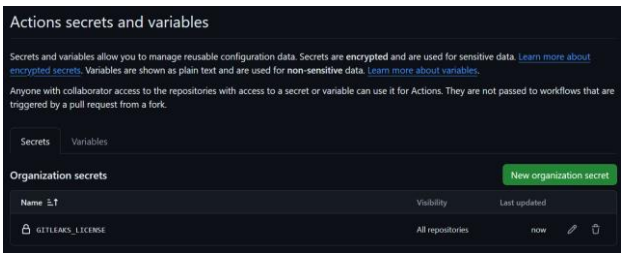


Figura 33: Secreto de Licencia GitLeaks
Fuente: elaboración propia

El fichero que define la GitHub Action se puede encontrar en el código adjunto a este trabajo en .github\workflows\GitLeaks.yml, pero es básicamente el facilitado por [49] con la salvedad de la ejecución de la acción de forma periódica. En la figura siguiente podemos ver el resultado de ejecutar un commit sin secretos:

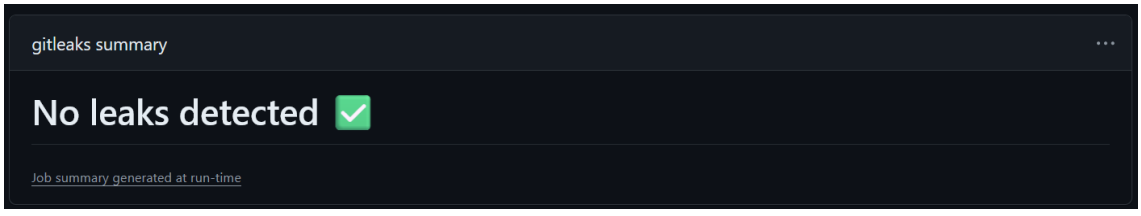


Figura 34: GitLeaks resultado de análisis sin secretos expuestos
Fuente: elaboración propia

Creo un fichero llamado .gitleaks.toml en la raíz del directorio para indicar a la GitHub Action las reglas a ejecutar. Con las primeras líneas, indico que quiero ejecutar las reglas por defecto y en [[rules]] añado una propia. Utilizo una expresión regular para indicar el patrón de textos a buscar y una serie de palabras clave.

Code Blame 17 lines (15 loc) · 362 Bytes

```
1
2   title = "TFM Configuración de Gitleaks"
3
4   [extend]
5   useDefault = true
6   [[rules]]
7   id = 'apex-rule-1'
8   description = "Regla para detección de secreto falso en apex"
9   regex = '''(?i)(?:apex)(?:.{0,10})(?:=){1}(?:.{0,5})[\"'`](.{4,120})[\"'`]'''
10  tags = ["apex", "secreto"]
11  secretGroup = 1
12  entropy = 3.5
13  keywords = [
14    "apex_auth",
15    "apex_password",
16    "apex_token",
17  ]
```

Figura 35: Fichero .gitleaks.toml

Fuente: elaboración propia

Aquí muestor el resultado de añadir a la clase BasicArithmetics.cls una serie de líneas susceptibles de ser detectadas:

desarrollador-tfm-jmgs Update BasicArithmetics.cls

Code Blame 26 lines (21 loc) · 888 Bytes ·

```
1   public with sharing class BasicArithmetics {
2
3       String Secret = '12345-12345-6789012-AFRT5-0000';
4       String apex_token_0 =  'AMFDGNIAEIRGE24345EFSDFS';
5       String apex_password = 'miPasswordSecreto';
6       String apex_token_1 =  'No';
7       String apex_corto = 'miPasswordSecreto';
8       String apex_demasiado_largo = 'mySecretPassword';
9       String apex_token_2 =  'DemasiadoLargoDemasiadoLargoDemasiadoLar
10
```

Figura 36: Secretos para la detección con GitLeaks

Fuente: elaboración propia

Tras el commit, el análisis muestra lo siguiente:

gitleaks summary

Gitleaks detected secrets

Rule ID	Commit	Secret URL	Start Line	Author	Date	Email	File
generic-api-key	bd5d1ae	View Secret	3	desarrollador-tfm-jmgs	2024-06-10T18:42:34Z	desarrollador.tfm.jmgs@gmail.com	force-app/main/default/classes/BasicArithmetics.cls
apex-rule-1	bd5d1ae	View Secret	4	desarrollador-tfm-jmgs	2024-06-10T18:42:34Z	desarrollador.tfm.jmgs@gmail.com	force-app/main/default/classes/BasicArithmetics.cls
apex-rule-1	bd5d1ae	View Secret	5	desarrollador-tfm-jmgs	2024-06-10T18:42:34Z	desarrollador.tfm.jmgs@gmail.com	force-app/main/default/classes/BasicArithmetics.cls
apex-rule-1	bd5d1ae	View Secret	7	desarrollador-tfm-jmgs	2024-06-10T18:42:34Z	desarrollador.tfm.jmgs@gmail.com	force-app/main/default/classes/BasicArithmetics.cls

Job summary generated at run-time

Figura 37: GitLeaks resultado de análisis con secretos expuestos

Fuente: elaboración propia

Se puede ver la id de la regla que ha detectado el secreto en la primera columna de la tabla de la figura anterior. La línea 3 ha sido detectada por una regla genérica, la 4, 5 y 7 cumplen con el patrón de la expresión regular introducida. La 6 no cumple porque el texto es demasiado corto, debería de ser de más de cuatro caracteres para saltar. La línea 8 no salta porque el nombre de la variable es demasiado largo y la línea 9 no salta porque el contenido de la cadena de texto es demasiado largo.

4.6. Configuración de SonarCloud/SonarLint

Para la creación de la cuenta de Sonar Cloud ver el anexo 4. El único punto importante para destacar es que definimos como nuevo código todo aquel código que haya cambiado en el último día con tal de agilizar los ejemplos.

Para la configuración de SonarLint en Visual Studio Code ver el anexo 5.

4.6.1. Ficheros para analizar con SonarCloud

Como ya hemos mencionado en el subapartado 3.6.2, SonarCloud cobra en función de las líneas a analizar. Por ello, voy a restringir el análisis del código a los ficheros que nos interesan.

Dicha configuración se realiza a nivel de proyecto, administración, configuración general, apartado de alcance del análisis. Aquí se puede seleccionar los ficheros a ignorar o a incluir mediante diferentes patrones y selectores. Dada la estructura de los proyectos de Salesforce, es más sencillo indicar lo que queremos incluir que excluir. Esta es la configuración para cada tipo de fichero que queremos analizar:

- Ficheros Apex: force-app/main/default/classes/*.cls
- Ficheros Javascript: force-app/main/default/lwc/*/*.js
- Ficheros HTML: force-app/main/default/lwc/*/*.html
- Ficheros CSS: force-app/main/default/lwc/*/*.css

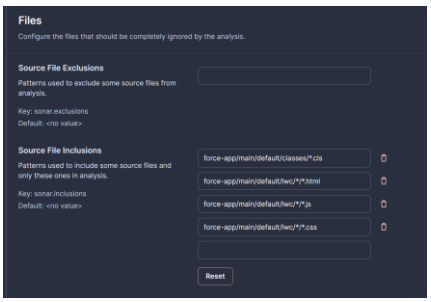


Figura 38: Inclusión de ficheros para SonarCloud
Fuente: elaboración propia

Por defecto, SonarCloud ya analiza la rama principal y las pull request. No queremos analizar los commits contra las ramas de desarrollo por tres motivos:

1. El análisis de secretos recae en GitLeaks.
2. Tenemos SonarLint para alertar de los errores en tiempo de desarrollo.
3. Para facilitar la colaboración. Los commits a las ramas de desarrollo son trabajo en progreso y es posible que se suban cambios que no sean definitivos para preservarlos temporalmente o abordarlos en grupo.

4.6.2. Quality Profiles de SonarCloud

Por lo general, utilizo la configuración base de SonarCloud. Para Apex, creo un nuevo perfil y activo todas las reglas posibles con el objetivo de mostrar la configuración a realizar en dicho escenario.

Los perfiles de calidad pueden crearse en su sección correspondiente del proyecto. Otorgo permiso de gestión al Líder técnico y funcional para que pueda gestionar las reglas. Dichas reglas se pueden activar o desactivar desde el propio perfil de forma individual o en masa como podemos observar en las siguientes figuras:

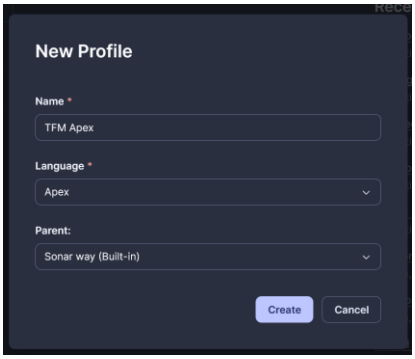


Figura 39: Nuevo perfil de calidad
Fuente: elaboración propia

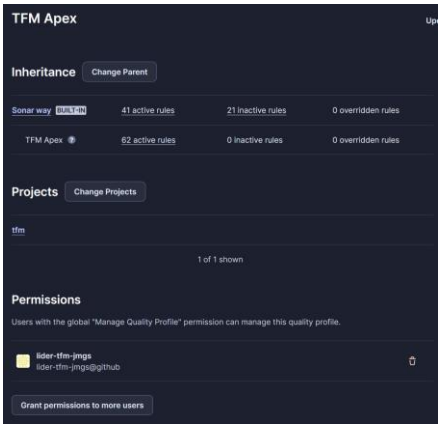


Figura 40: Configuración del perfil de calidad
Fuente: elaboración propia

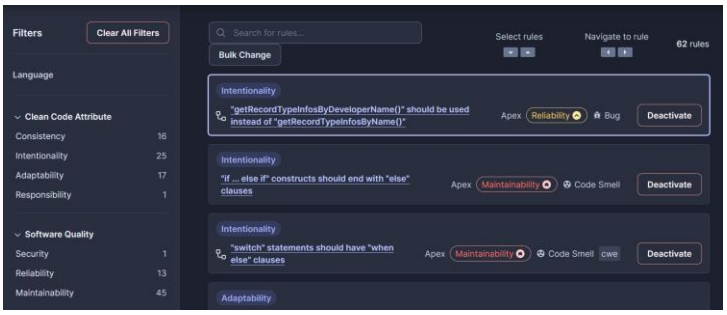


Figura 41: Activación y desactivación de reglas
Fuente: elaboración propia

4.6.3. Quality Gates de SonarCloud

Para la Quality Gate, voy a crear una nueva llamada TFM Quality Gate que va a heredar la mayoría de las reglas de la Quality Gate por defecto de SonarCloud. Las únicas diferencias serán:

- Quitar la cobertura de código, dado que los resultados los obtengo a partir de la GitHub Action que lanza el contenedor Docker de despliegue.
- Añadir bloqueante cuando tengamos más de 5 incidencias menores.
- Añadir bloqueante cuando tengamos más de 1 incidencia mayor.

Las reglas que aplicar recaerán en el código nuevo. La configuración de la Quality Gate queda como se muestra en la figura siguiente.

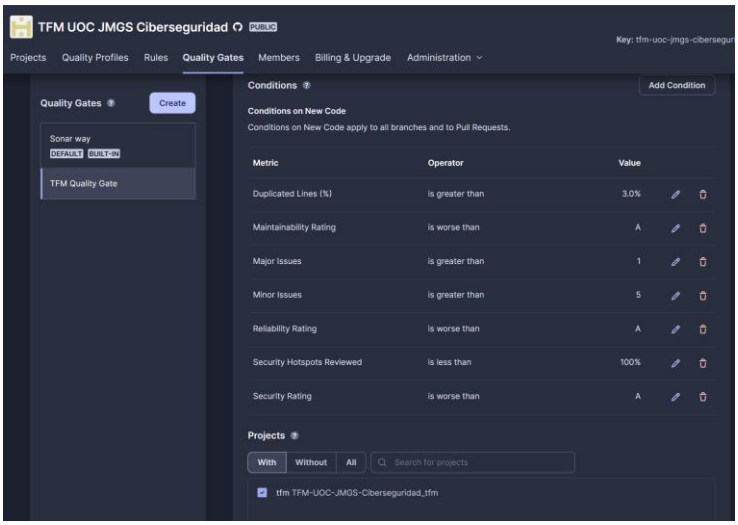


Figura 42: TFM Quality Gate de SonarCloud
Fuente: elaboración propia

Se puede observar el resultado de aplicar la Quality Gate en las siguientes figuras:

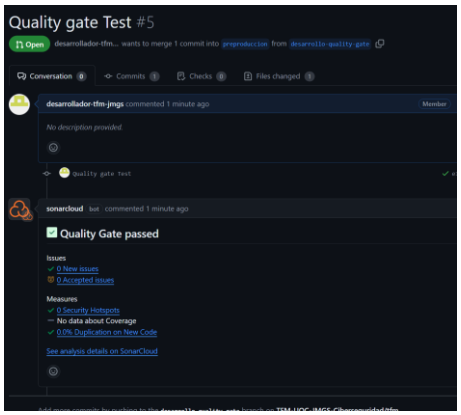


Figura 43: Quality Gate éxito

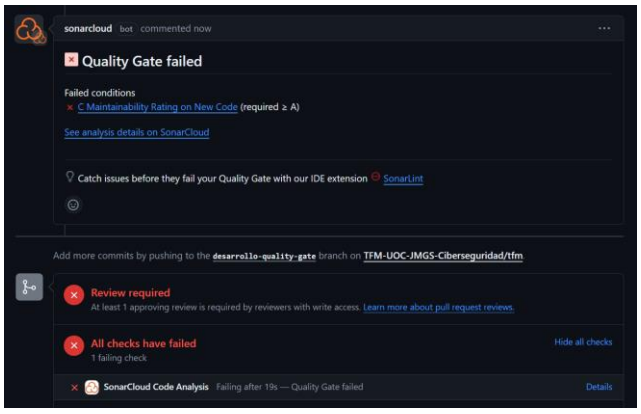


Figura 44: Quality Gate fallo

Para la primera figura he realizado un cambio mínimo del código actual. Para la segunda figura he modificado una función para tener múltiples variables sin uso y con valores repetidos.

4.6.4. Análisis disponibles en SonarCloud

Como comenté en el subapartado 2.4.1 y el apartado 3.6, sobre la interfaz de SonarCloud podemos ver con mayor detalle los errores y alertas detectados, así como el estado general del código de la rama main. El análisis está separado por código nuevo y código viejo.

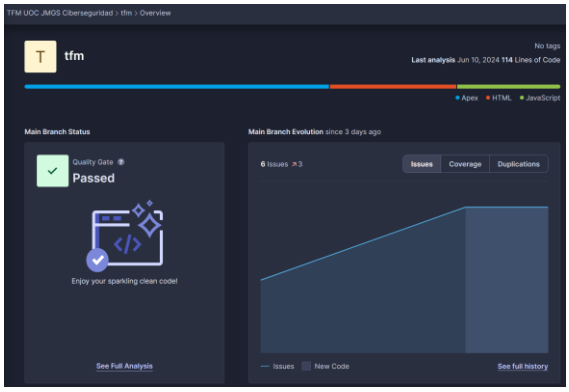


Figura 45: Información sobre estado general del código

Fuente: elaboración propia

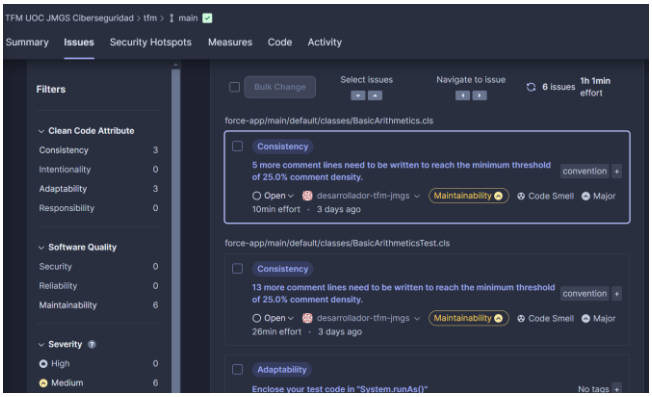


Figura 46: Alertas pendientes de revisar en SonarCloud

Fuente: elaboración propia

Se puede revisar el histórico de pull request y de los fallos detectados en ellas. También podemos consultar las reglas de las cuales derivan los errores y marcar los falsos positivos.

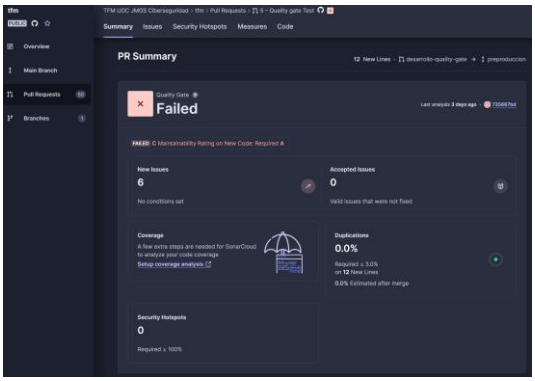


Figura 47: Información de fallo de Quality Gate en SonarCloud

Fuente: elaboración propia

4.7. Escaneo de contenedores

4.7.1. Análisis de las imágenes de Docker

Las imágenes que utilizar dentro de los contenedores de Dockers pueden analizarse con Docker Scout tras subir una nueva versión al repositorio de Docker Hub o en la herramienta de Docker Desktop [25] antes de subirlas al repositorio. Ver anexo 6 para el alta de la cuenta de Docker Hub.

El análisis nos permite ver las posibles vulnerabilidades de la imagen construida junto con las vulnerabilidades de sus dependencias. En la siguiente figura muestro uno de los resultados del análisis con Docker Scout.

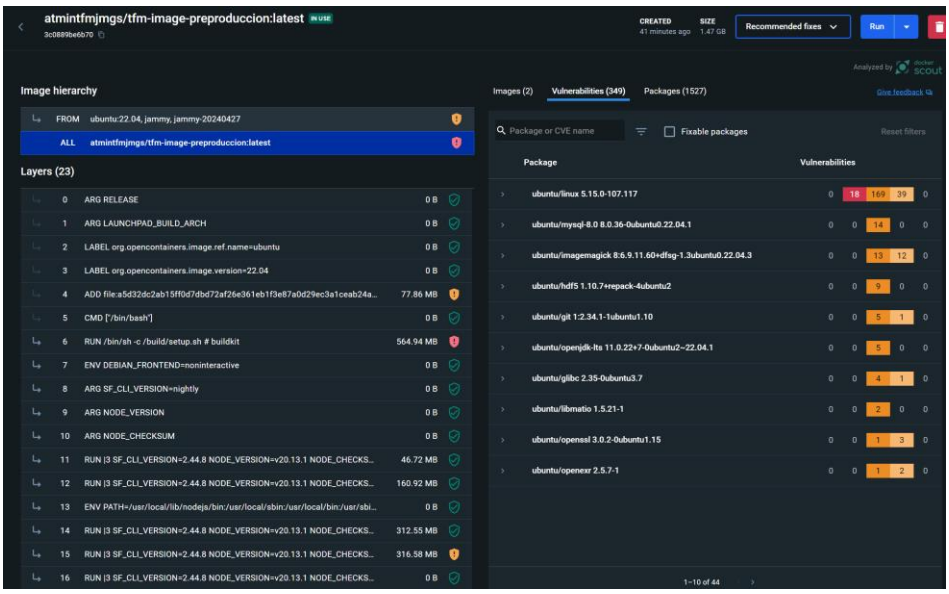


Figura 48: Resultados de Docker Scout
Fuente: elaboración propia

Podemos ver las vulnerabilidades conocidas de los diferentes elementos que conforman la imagen que hemos montado. La mayoría de ellos están relacionados con el sistema Ubuntu, que es el que utiliza la imagen que utilizamos de base, salesforce/cli, para ejecutar sus acciones. Para cada vulnerabilidad se nos informa de las acciones recomendadas.

Para el análisis en el repositorio de imágenes de Docker hay que seleccionar la siguiente opción en Docker Hub a nivel de repositorio:

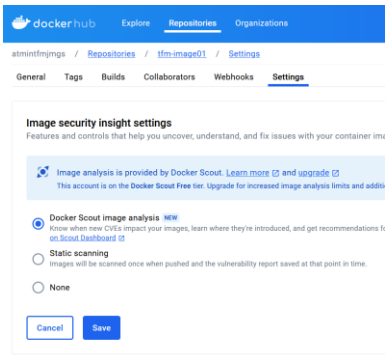


Figura 49: Activar de Docker Scout en el repositorio
Fuente: elaboración propia

4.8. Uso de ZAP

La herramienta ZAP tiene cuatro modos de ejecución:

- Modo seguro
- Modo protegido
- Modo estándar
- Modo ataque

Como comenté en el 3.8.1. Limitaciones de Salesforce, Salesforce impone serias limitaciones en el análisis DAST de su software. Dado que no tengo permiso para realizar un análisis activo, debo realizar uno pasivo.

Durante el análisis realizado en el subapartado 2.6.2 y en el diseño del subapartado 3.8.2 se me pasó por alto que los modos seguro y protegido no tienen capacidad de ejecución automática. El proceso de análisis pasivo requiere de navegación manual a través de las páginas de Salesforce. He investigado formas de forzar la navegación de forma automática dentro de un contenedor Docker sin éxito.

Por tanto, el análisis DAST deberá de ejecutarse fuera de los procesos automatizados en GitHub. He creado una GitHub Action [13], recordatorio-dast, para lanzar un recordatorio a modo de comentario en las pull request de preproducción y producción.

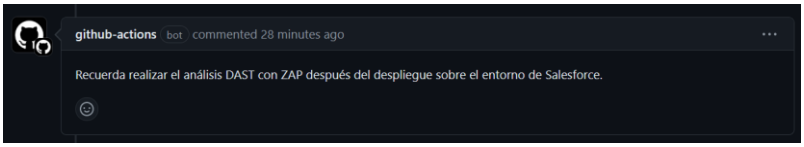


Figura 50: Recordatorio de ejecución de DAST con ZAP
Fuente: elaboración propia

Para la ejecución de ZAP uso la aplicación de escritorio para Windows. Con tal de poder lanzar el análisis sobre el navegador de Firefox he configurado el proxy en la definición de la red del navegador y referenciado el path al binario del navegador en la configuración de ZAP.

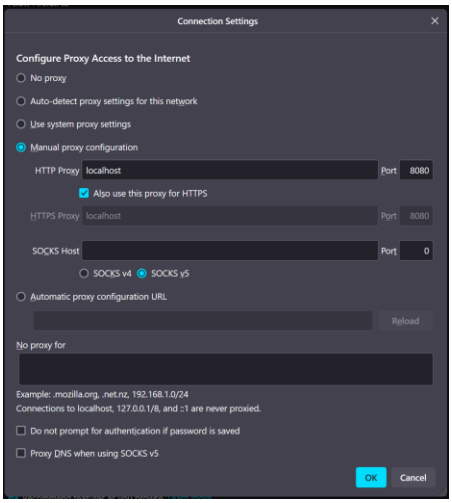


Figura 51: Configuración proxy para ZAP

Fuente: elaboración propia

Lanzó en análisis en modo seguro con la siguiente configuración de exploración manual sobre el entorno de dev:

Figura 52: ZAP Exploración Manual

Fuente: elaboración propia

La opción de 'Habilitar el HUD' permite superponer los controles que muestran el resultado del análisis sobre la interfaz del navegador como vemos en las siguientes figuras:

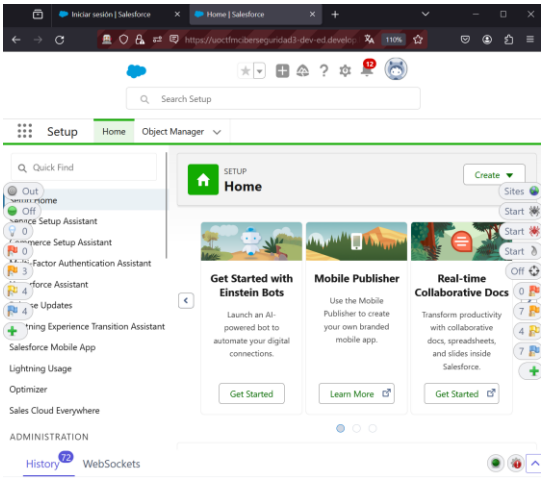


Figura 53: ZAP HUD

Fuente: elaboración propia

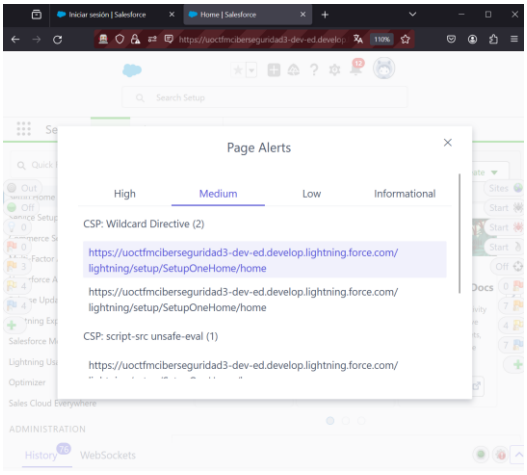


Figura 54: Alerta de página sobre Firefox

Fuente: elaboración propia

También podemos consultar la información sobre las alertas en la aplicación de ZAP donde las podemos ver en más detalle cómo podemos observar en la siguiente figura:

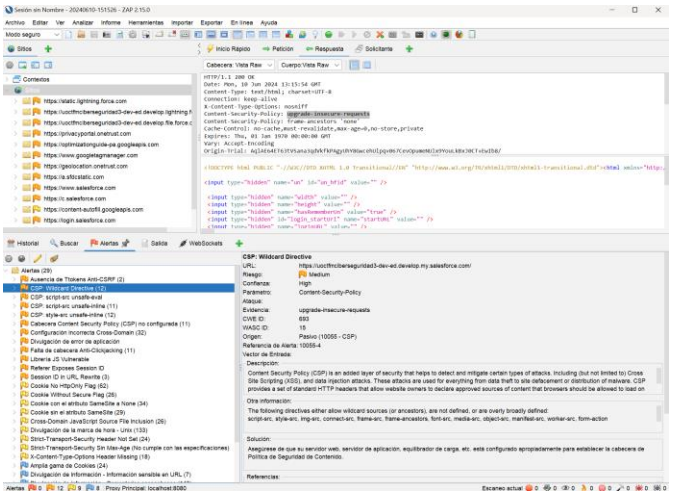


Figura 55: Análisis en la aplicación de ZAP
Fuente: elaboración propia

4.9. Cumplimiento Normativo

Siguiendo las instrucciones de OpenSearch Docker [54], doy de alta una cuenta para el usuario de administrador y utilizo el docker-compose.yml facilitado para lanzar OpenSearch Dashboards dentro de un contenedor Docker.

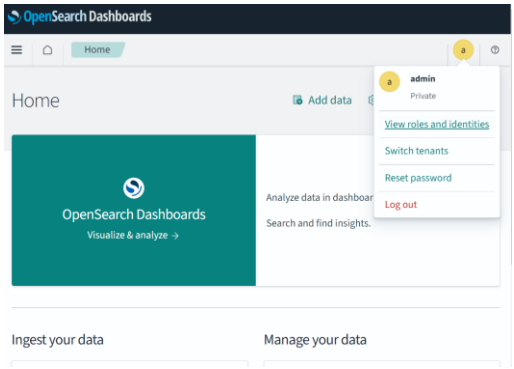


Figura 56: OpenSearch Dashboards página inicial
Fuente: elaboración propia

He intentado utilizar DataPrepper para recuperar los datos de Github sin éxito. OpenSearch Dashboards tiene varias opciones de integración, pero ninguna de ellas es GitHub y no he sido capaz de encontrar como añadir nuevas integraciones sin DataPrepper o OpenSearch base.

Por tanto, no he sido capaz de consolidar los datos de los Audits de GitHub y Salesforce para lanzar análisis sobre ellos. Ver el capítulo 5 sobre posibles alternativas.

Los Audit logs de GitHub los podemos encontrar, y descargar, en formato json o csv a nivel de enterprise los útlimos 180 días de actividad. Las cuentas gratuitas, con excepción de prueba de 30 días de Enterprise que estoy utilizando, no tienen esta opción.

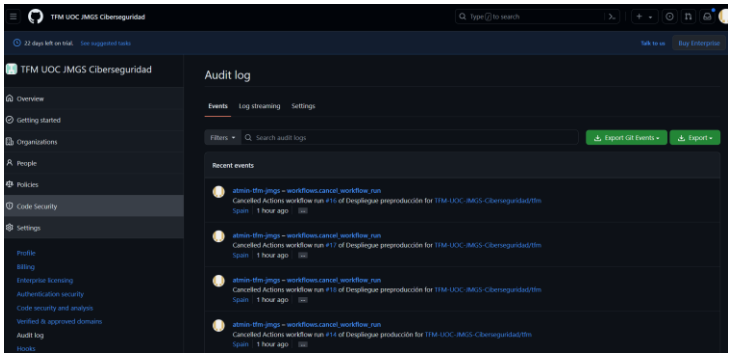


Figura 57: Audit Logs de GitHub
Fuente: elaboración propia

Los Audit Logs de Salesforce están limitados al estar trabajando con organizaciones de prueba, pero podemos consultar cambios de contraseña, intentos de log in exitosos y fallidos además de otros eventos a nivel de usuario. Podemos descargar esta información en formato csv para los últimos 6 meses. Con Salesforce Shield podemos ampliar la información a registrar y el tiempo a mantenerla.

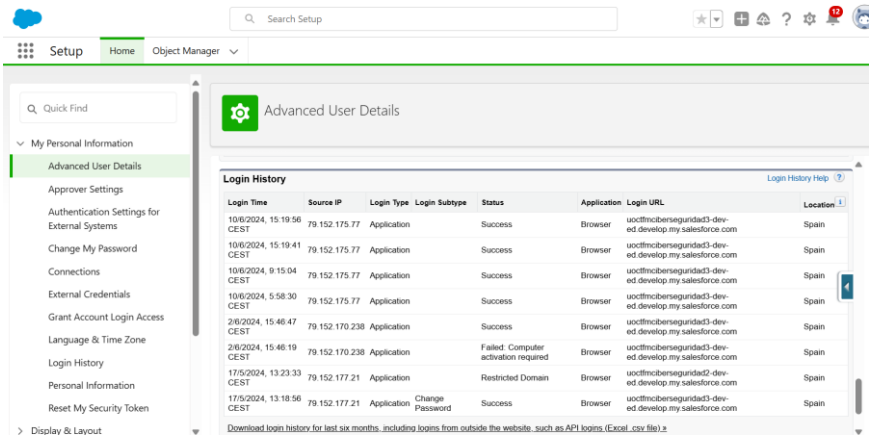


Figura 58: Audit Logs de Salesforce
Fuente: elaboración propia

4.10. Impacto sobre la sostenibilidad

4.10.1. Aspectos positivos

El grueso de la implementación del pipeline para este trabajo se ha realizado sobre los contenedores que ejecuta GitHub para sus Action. El impacto del análisis en busca de secretos, de los procesos de despliegue y de la ejecución de los test queda bajo el paraguas de GitHub.

Como menciono en el subapartado 2.1.1, GitHub es neutral con respecto a las emisiones de carbono y su aportación será negativa para 2030, por lo que las herramientas que estoy ejecutando sobre él serán neutrales también.

4.10.2. Aspectos negativos

Carecemos de información sobre el impacto ambiental de SonarCloud y dado que no podemos controlar la infraestructura sobre la que se ejecuta es muy difícil de estimar el impacto ambiental que su uso pueda tener. Una posibilidad sería utilizar SonarQube, la solución SonarCloud on-premise, y asegurarnos de que las máquinas sobre las que se ejecute el sistema son neutrales con respecto a las emisiones de carbono.

He utilizado Docker Hub para almacenar las imágenes de Docker que utilizo en la automatización de proceso. Tampoco tengo información sobre el coste ambiental del uso del repositorio de imágenes, pero nada nos impide utilizar un repositorio propio sobre el que se tendría un mayor control y capacidad de análisis de impacto energético.

5. Conclusiones y trabajos futuros

Este trabajo presenta una guía de implementación de un pipeline automatizado con entrega continua para soluciones Salesforce que utiliza herramientas de seguridad con tal de proteger el ciclo de desarrollo del software. No ha sido posible automatizar el análisis DAST debido a las restricciones de Salesforce.

Con respecto a la configuración del repositorio Git en GitHub, cabe destacar que me he visto obligado a utilizar una cuenta Enterprise para poder proteger el acceso a los secretos y bloquear los despliegues a productivo por parte de los desarrolladores. Además, para poder utilizar SonarCloud de forma gratuita he tenido que dejar los repositorios públicos. Ello implica que, para una implementación real, se requeriría pagar dos servicios, el de GitHub Enterprise y el de SonarCloud.

En cuanto a la detección de secretos, tras trabajar con GitLeaks, me ha quedado claro que este tipo de herramientas están pensadas para prevenir errores, no acciones intencionadas de introducir los secretos en abierto, dado que burlar los patrones de detección es relativamente fácil, pues introducir patrones demasiado genéricos darían falsos positivos en exceso.

En esta misma línea, el uso de las herramientas de análisis de código y del repositorio requieren de un equipo con la intención de aplicar y respetar las medidas de seguridad impuestas. DevSecOps es algo más que una configuración, es una cultura de equipo.

Queda pendiente la extracción de los Audit Logs para su representación en una herramienta de análisis de datos. Dado que solo tenemos dos sistemas que ofrecen Audit Logs, Salesforce y GitHub, una posibilidad sería utilizar el propio sistema de Salesforce para centralizar los Logs. Esto implicaría la creación de una solución en la plataforma para recuperar los datos y mostrarlos utilizando las herramientas disponibles para generar informes y gráficos. Para un análisis mucho más detallado, se podría utilizar el producto especializado de Salesforce Analytics.

Para aumentar la seguridad de la solución, sería interesante implementar un sistema de SSO con provisión de usuarios centralizada, establecer conexiones seguras utilizando certificados SSH y restringir las IPs de acceso a los diferentes recursos. En cuanto a la protección de secretos dentro de Salesforce, con una cuenta con Shield, sería interesante implementar la solución propuesta en el subapartado 3.4.2 o una similar.

6. Glosario

A.
AES-256: Algoritmo de encriptado de 256 bits de clave simétrica.

Apex: Lenguaje de programación propietario de Salesforce.

API: Application programing interface o interfaz de programación de aplicaciones.

AppExchange: Marketplace de Salesforce dónde se pueden obtener soluciones creadas por terceros para instalarlas sobre el software de Salesforce.

AWS: Amazon web services.

B.
Backend: término para indicar el conjunto de elementos de infraestructura, configuración y código que residen y se ejecutan en los servidores.

C.
CLI: Command line interface o interfaz de línea de comandos.

Cloud: Uso este término para referenciar las soluciones que no requieren de infraestructura propia de la empresa para funcionar.

D.
DAST: Dynamic application security testing o análisis dinámico de aplicaciones es una metodología a análisis de programas en tiempo de ejecución.

DevOps: Conjunto de prácticas y automatismos para la entrega de desarrollos de código.

DevSecOps: Conjunto de prácticas y automatismos para la entrega de desarrollos de código de forma segura.

DOM: Document Object Model. Estructura del documento html que da forma a una página web.

F.
Frontend: término para indicar el conjunto de elementos que residen y se ejecutan en el ordenador del usuario, normalmente a través de un navegador web o una aplicación.

H.
HIPAA: Health Insurance Portability and Accountability Act of 1996. Es una ley federal estadounidense que establece restricciones de privacidad sobre los datos de tipología médica.

I.
IDE: Integrated Development Environment o entorno de desarrollo integrado.

K.
KPI: Key Performance Indicator o indicador clave de rendimiento.

L.
LWC: Lightning Web Components. Framework de desarrollo de componentes web propietario de Salesforce.

O.
On-premise: Uso este término para referenciar las soluciones que requieren de infraestructura propia de la empresa para funcionar.

OWASP: Open Worldwide Application Security Project es una fundación sin ánimo de lucro que trabaja para mejorar la seguridad del software [59].

Q.
QA: Quality Assurance, control de calidad de la solución. Las pruebas de QA tienen como objetivo asegurar la calidad de la solución y que las funcionalidades de esta se comporten de la forma esperada antes de entregarlas a los usuarios finales para su validación final.

R.
Repositorio Git: Sistema de control de versiones distribuido.

RSA: Sistema criptográfico de clave pública.

S.
SAST: Static Application Security Test o análisis estático de aplicaciones es una metodología de análisis de programas a nivel de código para buscar vulnerabilidades en la fase de construcción.

SCA: Software Composition Analysis. Análisis del software para validar que las librerías y elementos de terceros que utiliza son seguros.

SOQL: Salesforce Object Query Language. Permite realizar búsquedas sobre la base de datos de Salesforce.

SOSL: Salesforce Object Search Language. Permite realizar búsquedas indexadas sobre múltiples tablas de la base de datos de Salesforce.

SSO: Single Sign On o inicio de sesión unificado es un procedimiento de autenticación que permite el acceso a múltiples sistemas mediante una única autenticación en uno de ellos.

U.

UAT: User Acceptance Test, pruebas de aceptación de usuario. Las pruebas de UAT tienen como objetivo validar que las funcionalidades de la solución implementada dan respuesta a las necesidades de los usuarios por los propios usuarios finales antes de su salida a productivo.

7. Bibliografía

[1] *Acerca de los ejecutores hospedados en GitHub* [en línea] [consulta: 14 de marzo de 2024]. Disponible en: https://docs.github.com/es/actions/using-github-hosted-runners/about-github-hosted-runners/about-github-hosted-runners

[2] *Amazon Sustainability The Cloud* [en línea] [consulta: 16 de marzo de 2024]. Disponible en: https://sustainability.aboutamazon.com/products-services/the-cloud?energyType=true

[3] *Apex Recipes* [consulta: 1 de junio de 2024]. Disponible en: https://github.com/trailheadapps/apex-recipes

[4] *AppExchange Security Review* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.packagingGuide.meta/packagingGuide/security_review_overview.htm

[5] *Atlassian Sustainability Report* [en línea] [consulta: 15 de marzo de 2024]. Disponible en: https://www.atlassian.com/company/corporate-social-responsibility/report

[6] *Autorabit* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://www.autorabit.com/

[7] *Bamboo* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://www.atlassian.com/software/bamboo

[8] *Bitbucket* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://bitbucket.org/

[9] *Características de AWS Secrets Manager* [en línea] [consulta: 16 de marzo de 2024]. Disponible en: https://aws.amazon.com/es/secrets-manager/features/

[10] *Chrome* [consulta: 26 de mayo de 2024]. Disponible en: https://www.google.es/chrome/

[11] *CIS Docker Benchmark* [en línea] [consulta: 24 de marzo de 2024]. Disponible en: https://www.cisecurity.org/benchmark/docker

[12] *Codeowners* [consulta: 7 de junio de 2024]. Disponible en: https://docs.github.com/es/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-code-owners

[13] *Comment on an issue* [consulta: 9 de junio de 2024]. Disponible en: https://github.com/actions/github-script#comment-on-an-issue

[14] *Compila flujos de trabajo potentes y automatizados* [en línea] [consulta: 15 de marzo de 2024]. Disponible en: https://www.atlassian.com/es/software/bitbucket/features/pipelines

[15] *Connected Apps* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://help.salesforce.com/s/articleView?id=sf.connected_app_overview.htm&type=5

[16] *Copado* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://www.copado.com/

[17] *Create a Connected App in your org* [consulta: 26 de mayo de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_auth_connected_app.htm

[18] *Create a Private Key and Self-Signed Digital Certificate* [consulta: 26 de mayo de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_auth_key_and_cert.htm

[19] *Creating a Docker container action* [consulta: 8 de junio de 2024]. Disponible en: https://docs.github.com/en/actions/creating-actions/creating-a-docker-container-action

[20] *Custom Metadata Types* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://help.salesforce.com/s/articleView?id=sf.custommetadatatypes_overview.htm&type=5

[21] *Custom Settings* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_customsettings.htm

[22] *Descripción general del código Apex* [en línea] [consulta: 10/03/2024]. Disponible en: https://help.salesforce.com/s/articleView?id=sf.code_about.htm&type=5

[23] *Developer Edition Sign up* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://developer.salesforce.com/signup

[24] *Docker Bench* [en línea] [consulta: 23 de marzo de 2024]. Disponible en: https://github.com/docker/docker-bench-security

[25] *Docker Desktop for Windows* [consulta: 8 de junio de 2024]. Disponible en: https://www.docker.com/products/docker-desktop/

[26] *Docker Scout* [en línea] [consulta: 23 de marzo de 2024]. Disponible en: https://docs.docker.com/scout

[27] *Docker Security Cheat Sheet* [en línea] [consulta: 23 de marzo de 2024]. Disponible en: https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html

[28] *Documentación de GitHub Actions* [en línea] [consulta: 14 de marzo de 2024]. Disponible en: https://docs.github.com/es/actions

[29] *Edge* [consulta: 26 de mayo de 2024]. Disponible en: https://www.microsoft.com/es-es/edge/download

[30] *Energy and power usage data in the cloud* [en línea] [consulta: 18 de marzo de 2024]. Disponible en: https://www.green-coding.io/blog/cloud-energy-usage-data/

[31] *Environment Hub* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.pkg1_dev.meta/pkg1_dev/environment_hub_intro.htm

[32] *Environmental sustainability at GitHub* [en línea] [consulta: 14de marzo de 2024]. Disponible en: https://github.blog/2021-04-22-environmental-sustainability-github/

[33] *Firefox* [consulta: 26 de mayo de 2024]. Disponible en: https://www.mozilla.org/es-ES/firefox/new/

[34] *Git* [consulta: 1 de junio de 2024]. Disponible en: https://git-scm.com/download/win

[35] *Git-secret* [en línea] [consulta: 16 de marzo de 2024]. Disponible en: https://sobolevn.me/git-secret/

[36] *GitGuardian* [en línea] [consulta: 17 de marzo de 2024]. Disponible en: https://www.gitguardian.com

[37] *GitHub* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://github.com/

[38] *GitHub pricing* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://github.com/pricing

[39] *GitLeaks* [en línea] [consulta: 17 de marzo de 2024]. Disponible en: https://gitleaks.io/

[40] *Gmail* [en línea] [consulta: 26 de mayo de 2024]. Disponible en: https://www.google.com/intl/es/gmail/about/

[41] *Green Software Foundation* [en línea] [consulta: 17 de marzo de 2024]. Disponible en: https://greensoftware.foundation/

[42] *How to Build a CI/CD Pipeline for Salesforce* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://www.salesforceben.com/how-to-build-a-ci-cd-pipeline-for-salesforce/

[43] *Informe sobre la cibercriminalidad en España* [en línea] [consulta: 11/03/2024]. Disponible en: https://www.interior.gob.es/opencms/export/sites/default/.galleries/galeria-de-prensa/documentos-y-multimedia/balances-e-informes/2022/Informe-Cibercriminalidad-2022.pdf

[44] *Introducing Lightning Web Components* [en línea] [consulta: 10/03/2024]. Disponible en: https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components

[45] *Invicti Tools* [en línea] [consulta: 26 de marzo de 2024]. Disponible en: https://www.invicti.com/

[46] *Java 22* [consulta: 26 de mayo de 2024]. Disponible en: https://www.oracle.com/java/technologies/downloads/

[47] *Jenkins* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://www.jenkins.io/

[48] *Jira* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://www.atlassian.com/software/jira

[49] *Licencia de GitLeaks* [consulta: 1 de junio de 2024]. Disponible en: https://gitleaks.io/products.html

[50] *Lightning Web Security* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://developer.salesforce.com/docs/platform/lwc/guide/security-lwsec-intro.html

[51] *Named Credentials and External Credentials* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://help.salesforce.com/s/articleView?id=sf.nc_named_creds_and_ext_creds.htm&type=5

[52] *Node 20* [consulta: 26 de mayo de 2024]. Disponible en: https://nodejs.org/en

[53] *OpenSearch* [en línea] [consulta: 08 de abril de 2024]. Disponible en: https://opensearch.org/

[54] *OpenSearch Docker* [consulta: 1 de junio de 2024]. Disponible en: https://opensearch.org/docs/latest/install-and-configure/install-opensearch/docker/

[55] *OpenSSL* [consulta: 26 de mayo de 2024]. Disponible en: https://www.openssl.org/

[56] *Opera* [en línea] [consulta: 26 de mayo de 2024]. Disponible en: https://www.opera.com/es/browsers

[57] *Opera GX* [en línea] [consulta: 26 de mayo de 2024]. Disponible en: https://www.opera.com/es/gx/gx-browser

[58] *Organize groups for your repositories* [en línea] [consulta: 15 de marzo de 2024]. Disponible en: https://support.atlassian.com/bitbucket-cloud/docs/organize-groups-for-your-repositories/

[59] *Owasp DevSecOps Guideline – v-0.2* [en línea] [consulta: 02 de marzo de 2024]. Disponible en: https://owasp.org/www-project-devsecops-guideline/latest/

[60] *Permisos de acceso en GitHub* [en línea] [consulta: 14 de marzo de 2024]. Disponible en: https://docs.github.com/es/get-started/learning-about-github/access-permissions-on-github

[61] *Revisar el registro de auditoría de tu organización* [en línea] [consulta: 14 de marzo de 2024]. Disponible en: https://docs.github.com/es/organizations/keeping-your-organization-secure/managing-security-settings-for-your-organization/reviewing-the-audit-log-for-your-organization

[62] *Salesforce* [en línea] [consulta: 10/03/2024]. Disponible en: https://www.salesforce.com/es/

[63] *Salesforce CLI* [consulta: 26 de mayo de 2024]. Disponible en: https://developer.salesforce.com/tools/salesforcecli

[64] *Salesforce Compliance* [en línea] [consulta: 08 de abril de 2024]. Disponible en: https://compliance.salesforce.com/en

[65] *Salesforce DX* [en línea] [consulta: 10/03/2024]. Disponible en: https://www.salesforce.com/products/platform/products/salesforce-dx/

[66] *Salesforce Extension Pack* [consulta: 26 de mayo de 2024]. Disponible en: https://marketplace.visualstudio.com/items?itemName=salesforce.salesforcedx-vscode

[67] *Sandobx Types and Templates* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://help.salesforce.com/s/articleView?id=sf.create_test_instance.htm&type=5

[68] *Semgrep* [en línea] [consulta: 21 de marzo de 2024]. Disponible en: https://semgrep.dev/

[69] *Semgrep Visual Studio Code extension* [en línea] [consulta: 21 de marzo de 2024]. Disponible en: https://semgrep.dev/docs/extensions/semgrep-vs-code/

[70] *Sonar* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://sonarcloud.io/

[71] *SonarCloud Documentation* [en línea] [consulta: 19 de marzo de 2024]. Disponible en: https://docs.sonarsource.com/sonarcloud/

[72] *SonarCloud Managing Permissions* [en línea] [consulta: 19 de marzo de 2024]. Disponible en: https://docs.sonarsource.com/sonarcloud/organizations/managing-permissions/

[73] *SonarCloud pricing* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://www.sonarsource.com/plans-and-pricing/

[74] *SonarCloud Sign Up* [consulta: 6 de junio de 2024]. Disponible en: https://www.sonarsource.com/products/sonarcloud/signup/

[75] *SonarLint* [en línea] [consulta: 19 de marzo de 2024]. Disponible en: https://www.sonarsource.com/products/sonarlint

[76] *SonarLint Connected Mode* [consulta: 6 de junio de 2024]. Disponible en: https://docs.sonarsource.com/sonarlint/vs-code/team-features/connected-mode-setup/

[77] *SonarLint Visual Studio Code Marketplace* [consulta: 6 de junio de 2024]. Disponible en: https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarlint-vscode

[78] *SOOS* [en línea] [consulta: 31 de marzo de 2024]. Disponible en: https://soos.io

[79] *Source Code Analysis Tools* [en línea] [consulta: 18 de marzo de 2024]. Disponible en: https://owasp.org/www-community/Source_Code_Analysis_Tools

[80] *Splunk* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://www.splunk.com/

[81] *Splunk global climate resilience and innovation strategy* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://www.splunk.com/en_us/global-impact/climate.html#innovation

[82] *Splunk User Behavior Analytics Guided* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://www.splunk.com/en_us/form/splunk-uba-interactive-demo.html

[83] *TC39* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://tc39.es/

[84] *Testing Apex* [en línea] [consulta: 09 de marzo de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing.htm

[85] *User and Role Access* [en línea] [consulta: 21 de marzo de 2024]. Disponible en: https://semgrep.dev/docs/deployment/user-management/

[86] *View and configure the audit log* [en línea] [consulta: 15 de marzo de 2024]. Disponible en: https://confluence.atlassian.com/bitbucketserver/view-and-configure-the-audit-log-776640417.html

[87] *Visual Studio Code* [consulta: 26 de mayo de 2024]. Disponible en: https://code.visualstudio.com/Download

[88] *Vulnerability Scanning Tools* [en línea] [consulta: 26 de marzo de 2024]. Disponible en: https://owasp.org/www-community/Vulnerability_Scanning_Tools

[89] *What's the difference between Classic Encryption and Shield Platform Encryption?* [en línea] [consulta: 04 de abril de 2024]. Disponible en: https://developer.salesforce.com/docs/atlas.en-us.securityImplGuide.meta/securityImplGuide/security_pe_vs_classic_encryption.htm

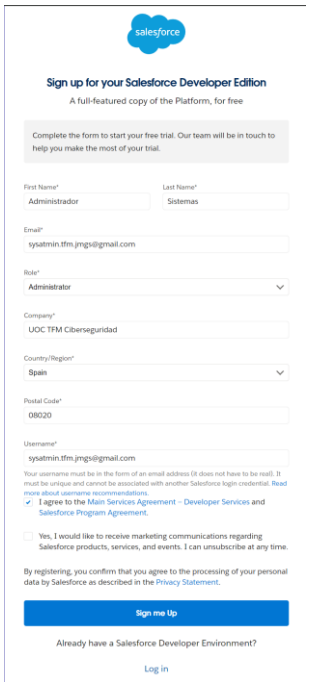
[90] *ZAP Marketplace* [en línea] [consulta: 31 de marzo de 2024]. Disponible en: https://www.zaproxy.org/addons/

[91] *Zed Attack Proxy (ZAP)* [en línea] [consulta: 29 de marzo de 2024]. Disponible en: https://www.zaproxy.org/

Anexos

1. Cuentas de Salesforce

Para el alta de las organizaciones se ha completado el formulario disponible en [23] utilizando el correo del administrador de sistemas.

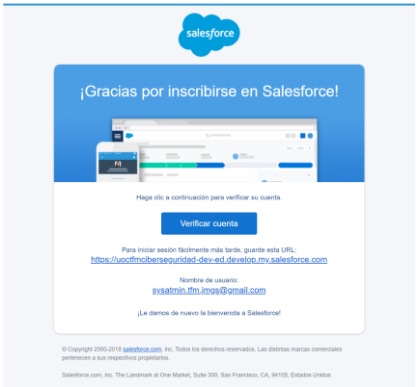


Anexo Figura 1: Formulario de alta DE

Fuente: Developer Edition Sign up [23]

La única diferencia para la creación de las cuentas ha radicado en la terminación del nombre de usuario, como indico en el apartado 4.2. El procedimiento es el mismo para las cuentas de desarrollo, preproducción y producción. Para el resto de los usuarios, el administrador de sistemas es quien los da de alta, pero los pasos para acceder son los mismos que los que muestro a continuación.

Una vez introducidos los datos, se confirma la cuenta desde el correo electrónico.

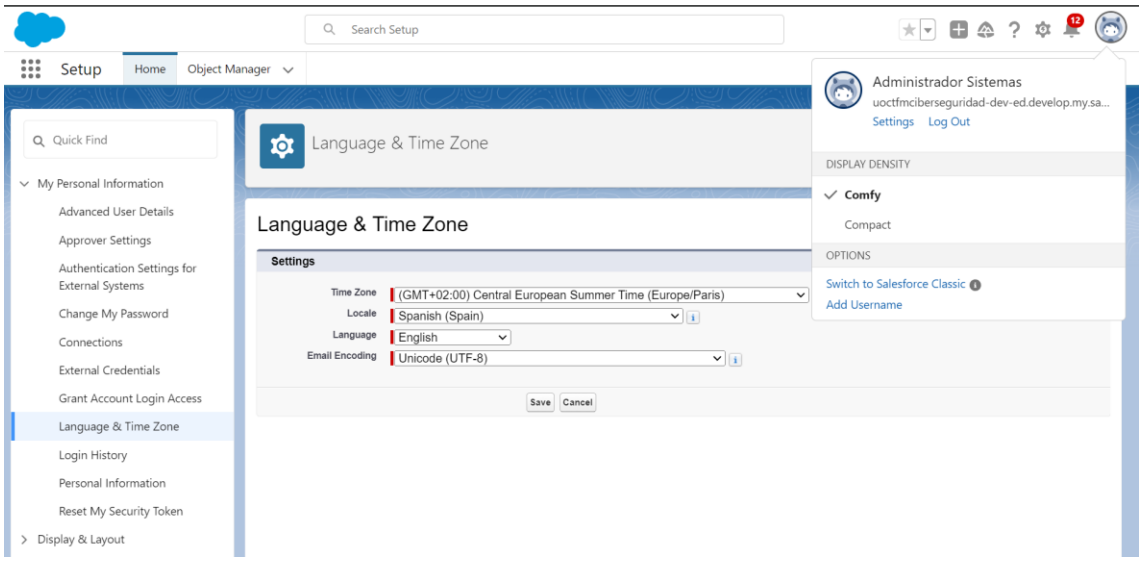


Anexo Figura 2: Verificación cuenta DE

Fuente: elaboración propia

Anexo Figura 3: Formulario de contraseña
Fuente: elaboración propia

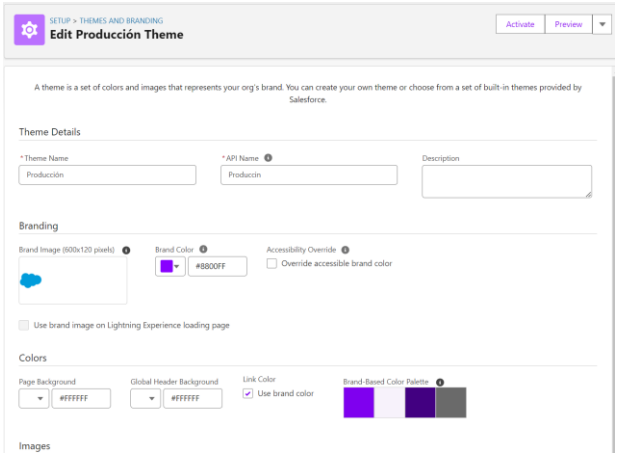
Tras completar el proceso de creación de contraseña, para facilitar los procesos de configuración, he configurado los usuarios para trabajar en inglés, dado que la documentación es más fiel al estado del sistema en este idioma.



Anexo Figura 4: Configuración del idioma DE
Fuente: elaboración propia

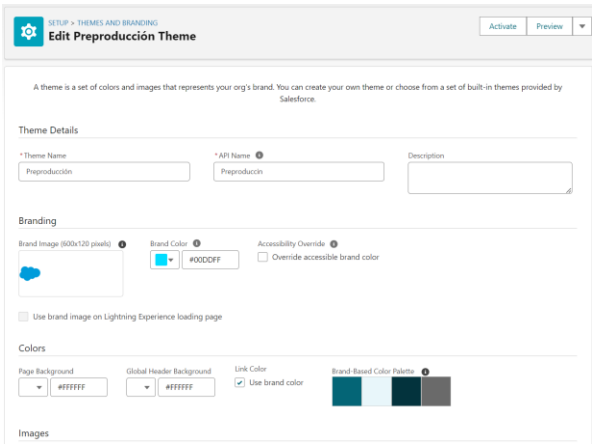
Para clarificar en las capturas y videos, altero la paleta de colores de las organizaciones para que sea más fácil de distinguir en que entorno me encuentro en cada momento.

DE Producción



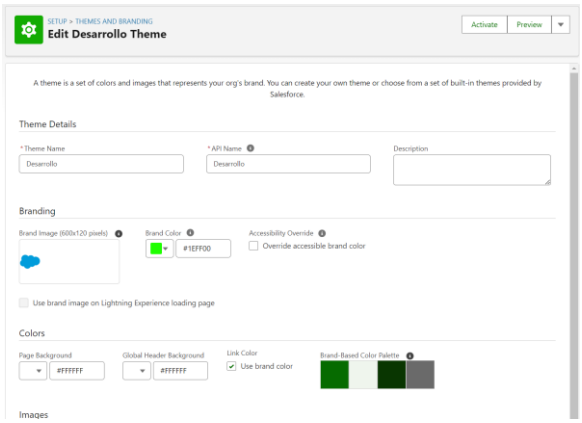
Anexo Figura 5: Tema DE Producción
Fuente: elaboración propia

DE Preproducción



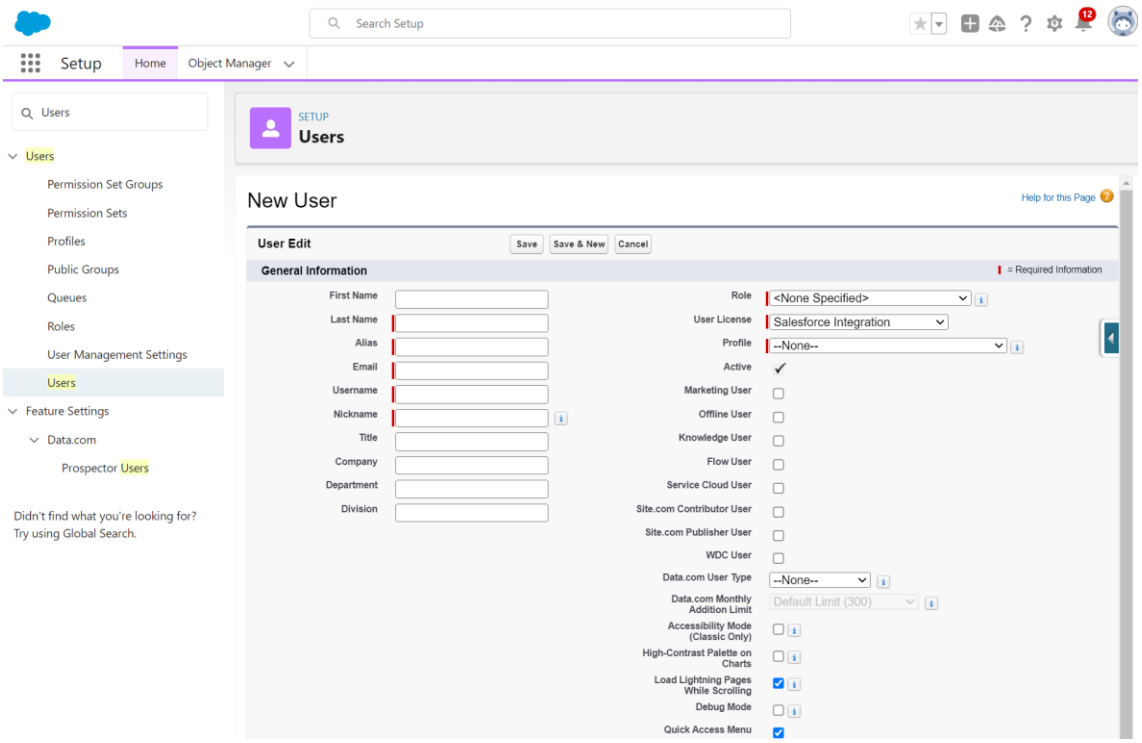
Anexo Figura 6: Tema DE Preproducción
Fuente: elaboración propia

DE Desarrollo



Anexo Figura 7: Tema DE Desarrollo
Fuente: elaboración propia

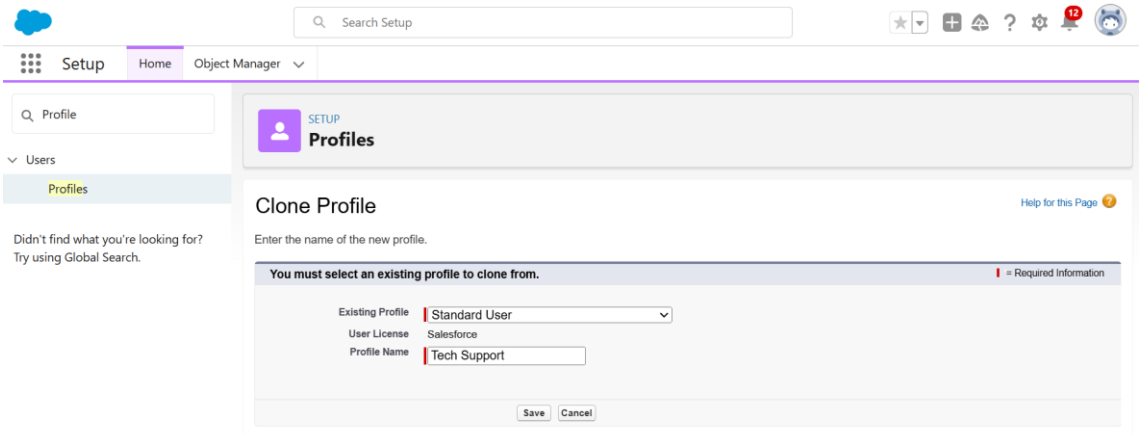
El administrador de sistemas utiliza el formulario de la siguiente figura para dar de alta a los usuarios. El formulario consta de más campos, pero los principales son los aquí mostrados. No modifico la configuración por defecto de requerimientos de contraseñas y del uso de MFA.



Anexo Figura 8: Formulario de alta de usuario Salesforce

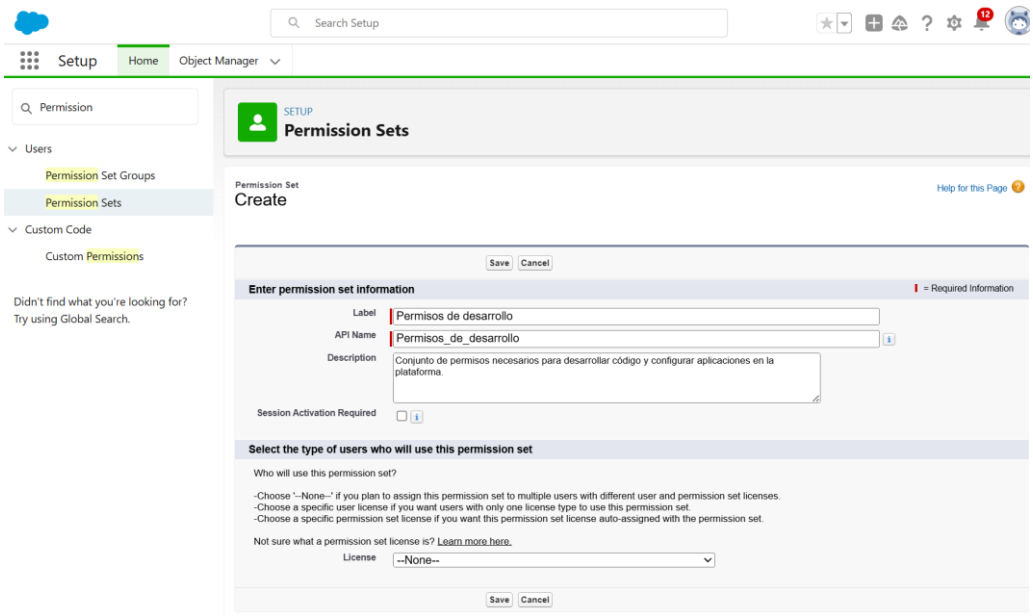
Fuente: elaboración propia

Para el perfil, he creado un nuevo perfil llamado Tech Support que utiliza como base el perfil de usuario estándar de Salesforce. Para que el usuario de desarrollo tenga capacidad de modificar la plataforma y desarrollar, le daré permisos adicionales mediante un Permission Set llamado Permisos de desarrollo.



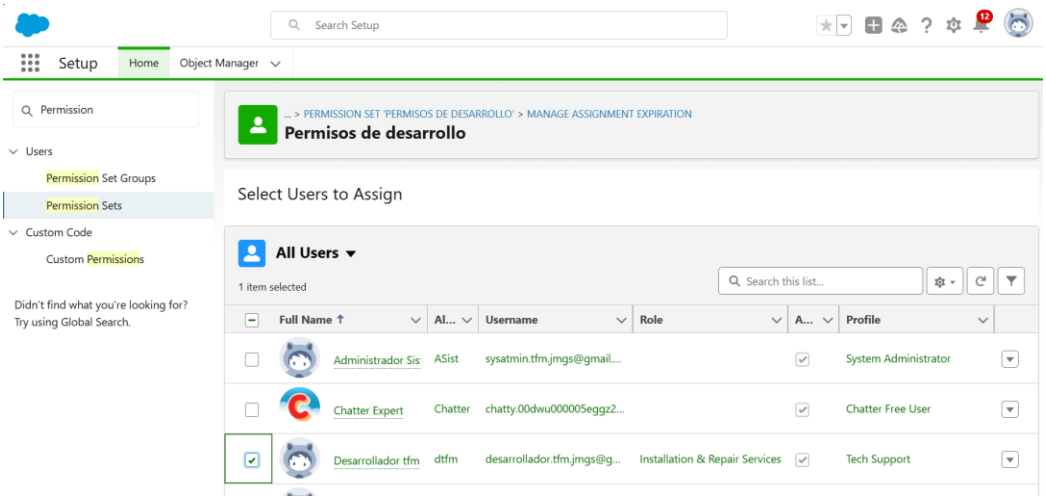
Anexo Figura 9: Creación del perfil de Tech Support

Fuente: elaboración propia



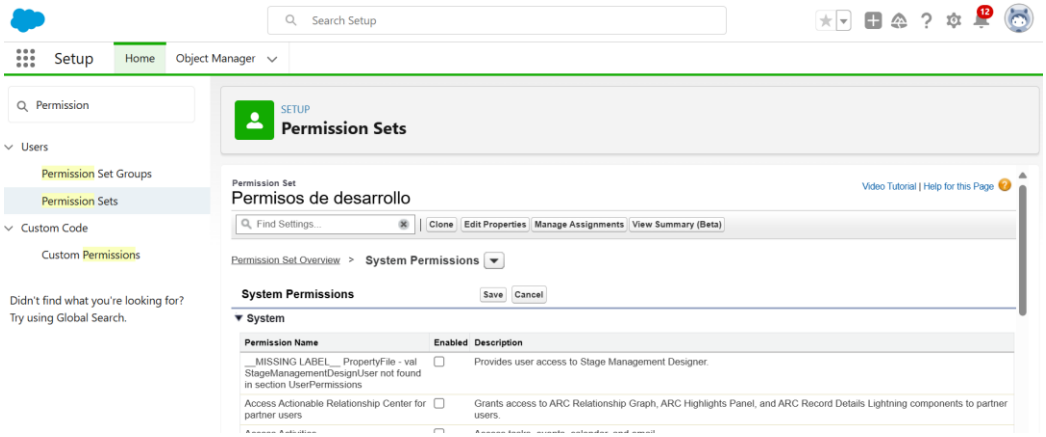
Anexo Figura 10: Creación de permisos de desarrollo

Fuente: elaboración propia



Anexo Figura 11: Asignación de permisos de desarrollo

Fuente: elaboración propia

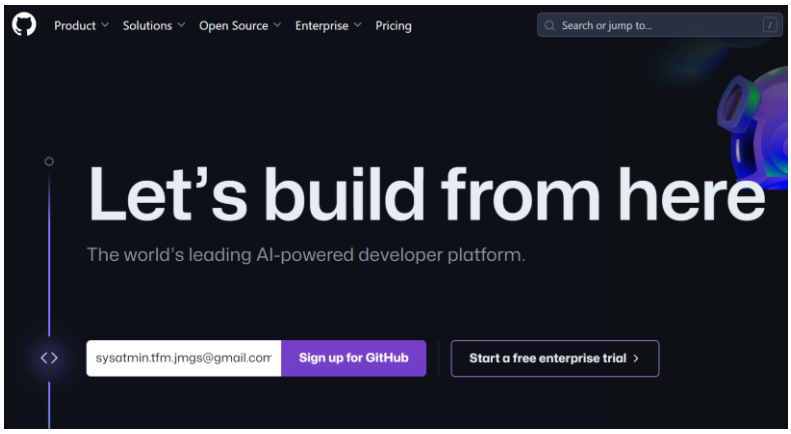


Anexo Figura 12: Configuración de los permisos de desarrollo

Fuente: elaboración propia

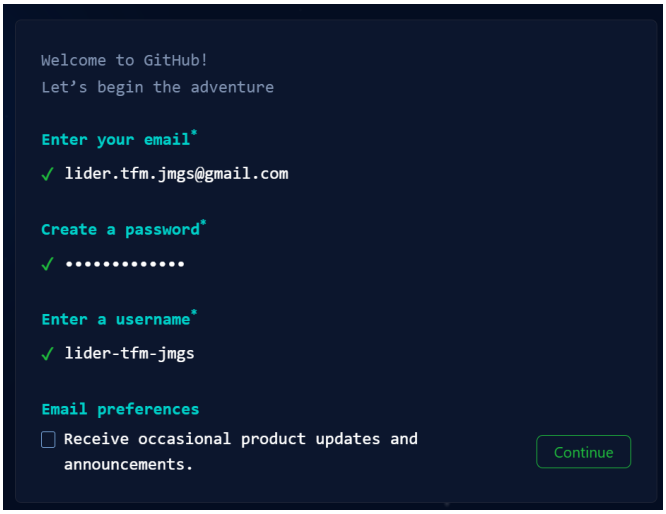
2. Cuentas de GitHub

Las cuentas de Github se crean utilizando los correos de Gmail de los usuarios entrando en la página principal de Github [37].



Anexo Figura 13: Sign up for GitHub
Fuente: elaboración propia

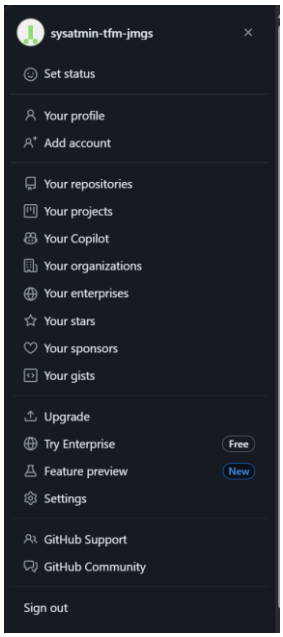
Introducimos el correo electrónico, el nombre de usuario y la contraseña. Tras validar la cuenta con el código que GitHub manda al correo electrónico, ya podemos entrar.



Anexo Figura 14: Formulario de alta de GitHub
Fuente: elaboración propia

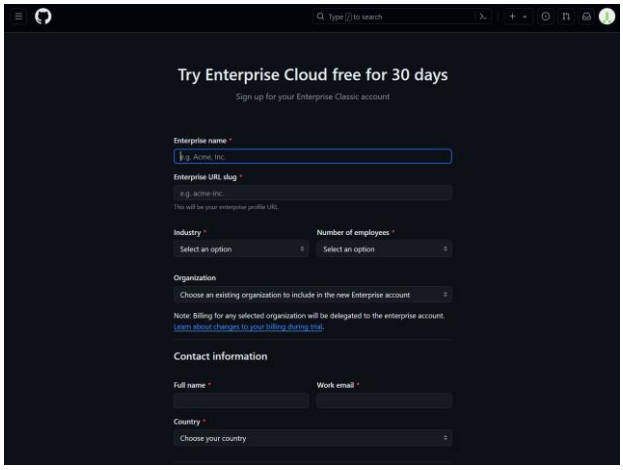
Elijo saltar la personalización para todas las cuentas.

El siguiente paso es solicitar una cuenta Enterprise de treinta días de prueba. Para ello accedemos con nuestro usuario de administrador de sistemas y nos dirigimos a la sección del perfil del usuario.



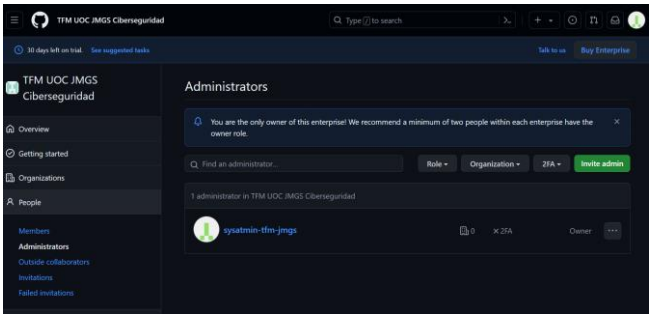
Anexo Figura 15: Try Enterprise
Fuente: elaboración propia

Seleccionamos la opción para probar Enterprise y elegimos Enterprise Cloud.



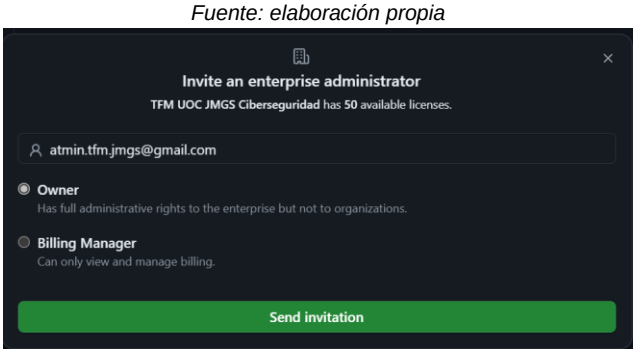
Anexo Figura 16: Formulario GitHub Enterprise
Fuente: elaboración propia

Rellenamos el formulario con los datos requeridos y accedemos a la cuenta de Enterprise. Procedemos a invitar al administrador.



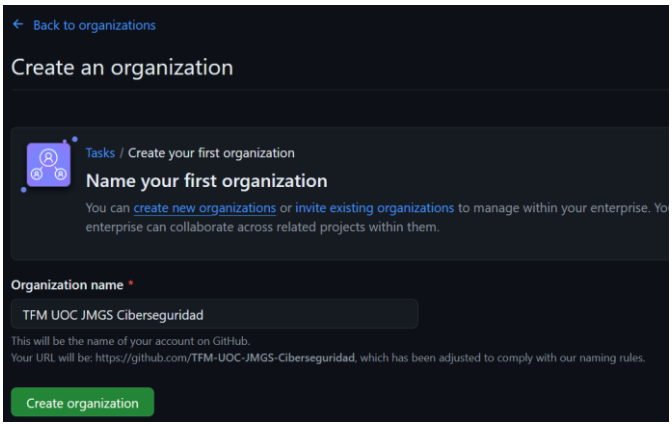
Anexo Figura 17: Invitación al administrador

Fuente: elaboración propia



Anexo Figura 18: Rol del administrador
Fuente: elaboración propia

Ahora, utilizando el usuario de administrador, creamos la organización que irá asociada a la cuenta de Enterprise.



Anexo Figura 19: Creación de la organización
Fuente: elaboración propia

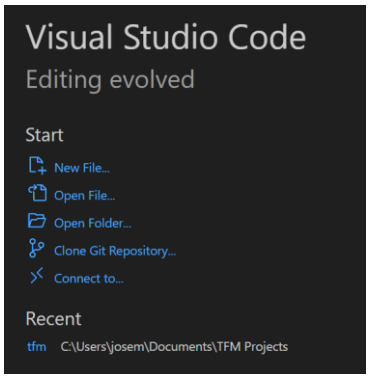
Configuramos los permisos generales para los miembros.

- Retiro los permisos de creación de repositorios.
- Retiro el permiso de invitar a miembros externos a la organización a los repositorios de la organización.
- Dejo la creación de páginas en privado.
- Desactivo las peticiones de integración de externos.
- Quito los permisos de modificación y borrado de los repositorios existentes.
- Dejo la creación de equipos e insights.

Invito al administrador de sistemas como owner de la organización para que pueda realizar las acciones necesarias de configuración con otras cuentas como la de SonarCloud.

3. Subida del código inicial

Instalo la versión 2.45.2 64-bit para Windows de Git [34] con todos los valores dados por el instalador por defecto. Visual Studio Code detecta la instalación de Git y nos permite interactuar con los repositorios de GitHub a través de su interfaz como vemos en la siguiente figura:



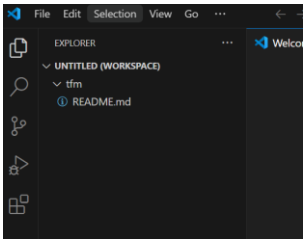
Anexo Figura 20: Opción de clonar un repositorio Git
Fuente: elaboración propia

Informamos a Git de quienes somos:

```
PS C:\Users\josem\Documents\TFM Projects\tfm> git config --global user.email "desarrollador.tfm.jmgs@gmail.com"
PS C:\Users\josem\Documents\TFM Projects\tfm> git config --global user.name "desarrollador-tfm-jmgs"
```

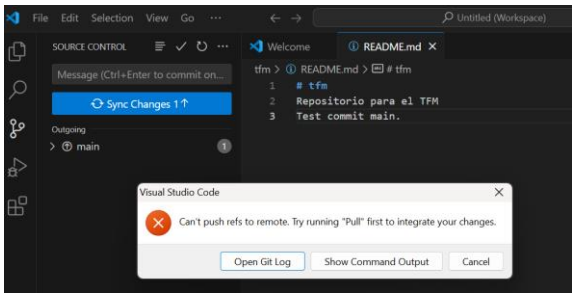
Anexo Figura 21: Información para Git del Desarrollador
Fuente: elaboración propia

Seleccionamos Clone Git Repository, nos autenticamos como desarrollador para poder acceder al repositorio y procedemos a clonar el repositorio de tfm.



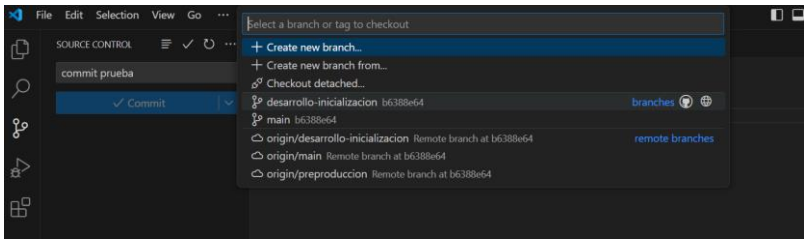
Anexo Figura 22: Repositorio clonado
Fuente: elaboración propia

Una vez clonado el repositorio, hacemos una validación rápida para comprobar que realmente no podemos hacer commits sin merge contra la rama Main:

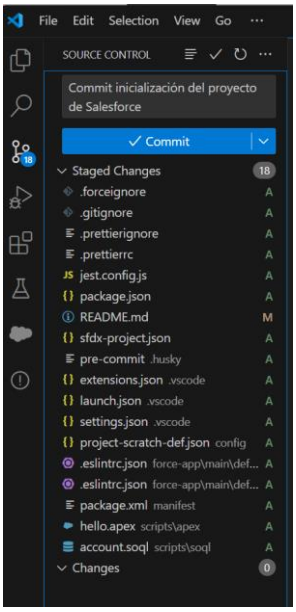


Anexo Figura 23: Rama Main protegida
Fuente: elaboración propia

Cambiamos la rama a la de desarrollo y nos traemos la base del proyecto de Salesforce para la primera subida y incluimos los cambios para realizar el commit sobre la rama de desarrollo.



Anexo Figura 24: Cambio de rama a desarrollo
Fuente: elaboración propia

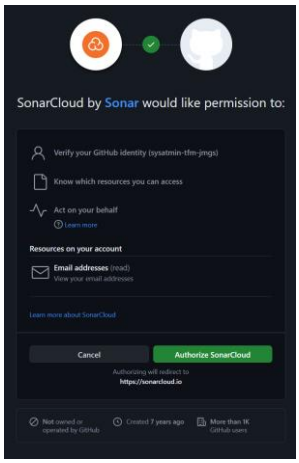


Anexo Figura 25: Commit de inicialización
Fuente: elaboración propia

Añadimos algunas clases de Apex y componentes elaborados a partir de los ejemplos del repositorio de Salesforce [3].

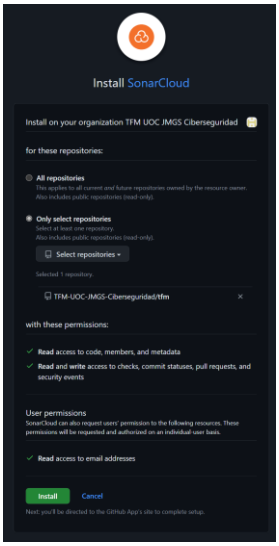
4. Cuenta de SonarCloud

Nos damos de alta en SonarCloud [74] utilizando la cuenta de GitHub del administrador de sistemas.



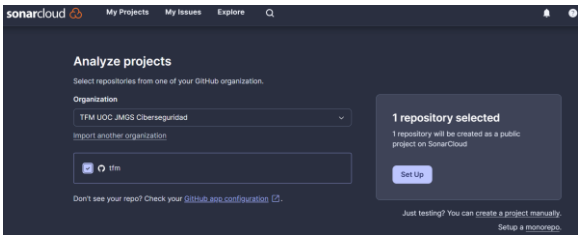
Anexo Figura 26: SonarCloud alta con Github
Fuente: elaboración propia

Seleccionamos la opción para aplicar SonarCloud sobre nuestra organización de GitHub y procedemos con la instalación:



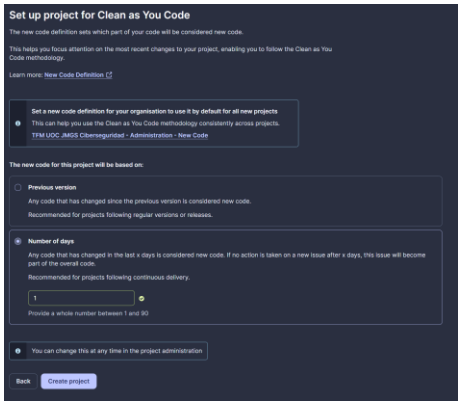
Anexo Figura 27: Instalación SonarCloud
Fuente: elaboración propia

Al crear la organización de SonarCloud a partir de la organización de GitHub, nos traemos los miembros de GitHub a SonarCloud. Selecciono la opción de la cuenta gratuita de SonarCloud, sin cambiar los nombres asignados por defecto a la organización. Seleccionamos el repositorio a configurar:



Anexo Figura 28: Análisis SonarCloud selección repositorio
Fuente: elaboración propia

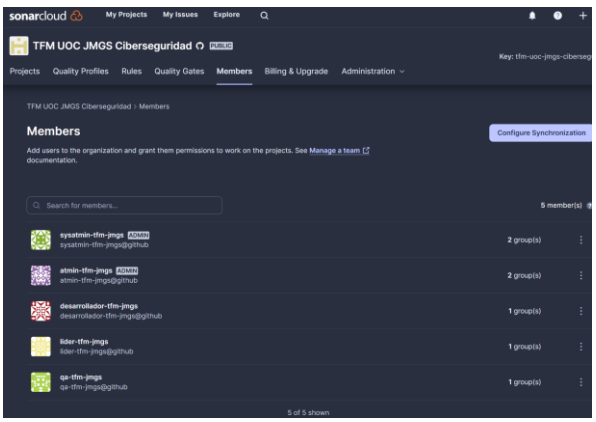
Con respecto a la política de Clean As You Code de Sonar Cloud, me decanto por la que considera como código nuevo cualquier código que haya cambiado en el último día. Idealmente, el margen debería adaptarse a las necesidades del proyecto, pero un día me permite realizar pruebas de forma más eficiente.



Anexo Figura 29: Configuración base del proyecto de SonarCloud
Fuente: elaboración propia

SonarCloud procede a analizar el código del repositorio y nos da un resultado inicial. A partir de aquí, se puede configurar SonarCloud según nuestras necesidades.

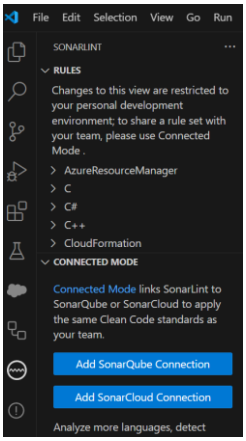
El resto de los usuarios pueden acceder a SonarCloud utilizando la cuenta de GitHub y tendrán la opción de entrar en el proyecto creado por el administrador de sistemas con los roles asociados de GitHub dado que SonarCloud tiene la opción de sincronizar las cuentas. Esta sincronización puede desactivarse.



Anexo Figura 30: SonarCloud miembros
Fuente: elaboración propia

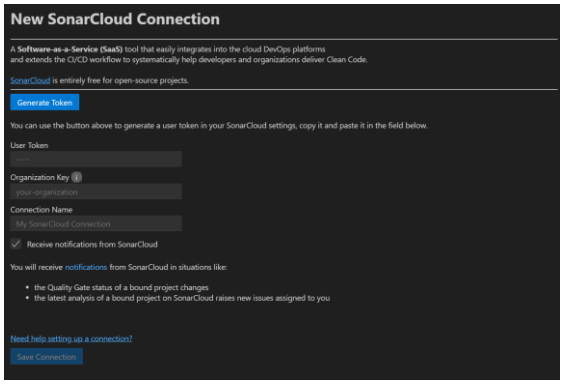
5. SonarLint en Visual Studio Code

Para la configuración de SonarLint en Visual Studio Code necesitamos instalar la extensión de SonarLint [77] y seguir los pasos indicados en Connected Mode para conectar con la instancia de SonarCloud. Encontramos la opción de conexión:



Anexo Figura 31: Añadir conexión SonarCloud para SonarLint

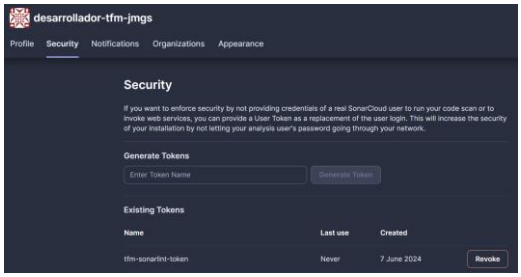
Fuente: elaboración propia



Anexo Figura 32: Nueva conexión SonarCloud

Fuente: elaboración propia

Generamos el token de usuario en la cuenta de SonarCloud del desarrollador e introducimos los datos solicitados.



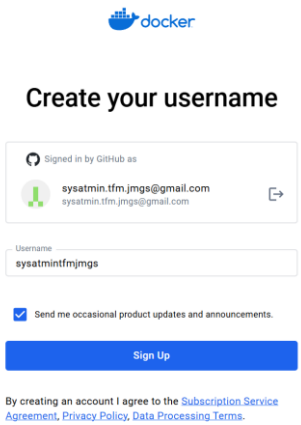
Anexo Figura 33: Token SonarLint

Fuente: elaboración propia

En caso de error, revisar el path de Nodejs en la configuración de Visual Studio Code para SonarLint, limpiar la cache del programa y reiniciarlo.

6. Cuenta de Docker Hub

Utilizo la cuenta de GitHub para darme de alta en Docker Hub [25] con el administrador de sistemas y otra con el administrador.



Anexo Figura 34: Alta Cuenta Docker Hub
Fuente: elaboración propia

Utilizaré la cuenta del administrador porque los equipos de colaboración son de pago y no puedo hacer la configuración de los roles.

7. Lista de programas y herramientas utilizadas

C.

- Chrome [10]

D.

- Docker Desktop [25]

E.

- Edge [29]

F.

- Firefox [33]

G.

- Git [34]
- GitHub [38]
- GitLeaks [39]
- Gmail [40]

J.

- Java 17 y 22 [46]

M.

- macOS Catalina
- Microsoft Word Modo Compatible

N.

- Node [52]

O.

- OpenSearch [54]
- Openssl [55]
- Opera [56]
- Opera GX [57]

P.

- Paquete de extensiones de Salesforce [66]

S.

- Salesforce CLI [63]
- Salesforce Developer Edition [23]
- SonarCloud [70]
- SonarLint [75]

V.

- Visual Studio Code [87]

W.

- Windows 11

Z.

- ZAP 2.15.0 [91]

8. Ficheros adjuntos

Las carpetas de tfm-image-preproduccion y tfm-image-produccion contienen el Dockerfile y el sf-dex-commands.sh que he utilizado para generar las imágenes Docker que luego referencio en las GitHub Actions del repositorio. Los ficheros de tfm-image-preproduccion y tfm-image-produccion son los mismos, pero los tengo separados para facilitar divergencias futuras en caso necesario.

La carpeta de tfm-main contiene los ficheros de la rama main del repositorio. Las principales rutas y ficheros a tener en cuenta:

- .github/workflows
    - Ficheros yml para definir las GitHub Actions que ejecuta el repositorio.
- .github/CODEOWNERS
    - Fichero para definir a los propietarios de los ficheros contenidos en el repositorio.
- Docker-actions
    - Contiene carpetas con los dicheros Dockerfile y action que inicializan los contenedores para ejecutar las imágenes de despliegue del código.
- force-app/main/default
    - Contiene en código de Salesforce.
- manifest
    - Contiene el fichero que indica a las instrucciones de despliegue que hay que desplegar de la ruta force-app/main/default.
- .gitleaks
    - Fichero para definir las reglas del análisis de GitLeaks.